



Numéro National de Thèse : 2019LYSEN054

THÈSE DE DOCTORAT DE L'UNIVERSITÉ DE LYON

opérée au sein de
l'École Normale Supérieure de Lyon

École Doctorale N° 512
École Doctorale en Informatique et Mathématiques de Lyon

Spécialité de doctorat : Informatique

Soutenue publiquement le 17/10/2019, par :
Aurore ALCOLEI

Jeux concurrents enrichis : témoins pour les preuves et les ressources

Enriched concurrent games : witnesses for proofs and resource analysis

Devant le jury composé de :

GHICA Dan, Reader, University of Birmingham (Royaume-Uni)

MILLER Dale, Directeur de Recherche, INRIA

FAGGIAN Claudia, Chargée de Recherche, CNRS

KESNER Delia, Professeure, IRIF et Université Paris Diderot

LAURENT Olivier, Directeur de Recherche, CNRS et ENS de Lyon

CLAIRAMBAULT Pierre, Chargé de Recherche, CNRS et ENS de Lyon

WINSKEL Glynn, Professeur, University of Cambridge

Rapporteur

Rapporteur

Examinatrice

Examinatrice

Directeur de thèse

Co-encadrant

Co-directeur

Résumé

La sémantique des jeux est une sémantique dénotationnelle centrée sur l'interaction : preuves et programmes y sont représentés par des stratégies modélisant, par le flot d'exécution, leur manière de réagir à leur environnement. Malgré cette présentation intensionnelle, les sémantiques de jeux ne suffisent pas à capturer certaines informations calculatoires annexes au flot d'exécution telles que, par exemple, la production de témoins en logique du premier ordre ou la consommation de ressources dans les langages de programmation. Dans cette thèse, nous proposons un cadre général pour enrichir les sémantiques de jeux concurrentes avec des annotations permettant de garder trace de ces informations. Ces enrichissements sont donc aussi l'occasion d'investiguer plus avant l'expressivité des modèles de jeux concurrents.

Nous construisons d'abord un modèle de jeux concurrent dans lequel les coups joueurs d'une stratégie sont annotés par les termes d'une théorie (in)équationnelle. Cette théorie est un paramètre de notre modèle et les annotations permettent de refléter de manière compacte des informations d'exécution n'ayant pas d'influence sur le flot d'exécution. Nous montrons que le modèle ainsi construit préserve la structure catégorique compacte fermée du modèle sans annotations.

Nous explorons ensuite l'expressivité des modèles annotés et présentons deux interprétations nouvelles en sémantique des preuves et des programmes : la première interprète les preuves de la logique classique du premier ordre par des stratégies concurrentes avec échange de témoins, donnant une version compositionnelle au théorème de Herbrand ; la seconde permet de refléter les aspects quantitatifs liés à la consommation de ressources telles que le temps, dans l'exécution de programmes concurrents d'ordre supérieur avec mémoire partagée. Ces sémantiques mettent en avant la portée des informations capturées dans leurs calculs respectifs, à savoir qu'elles n'influencent pas leur flot d'exécution mais sont influencées par ce dernier.

Abstract

Game semantics is an interactive denotational semantics: a denotation specifies the behaviour of a term/proof with respect to its environment. As such it is one of the most intensional model available in the Curry-Howard community. Despite their intensional perspective, game models still omit a number of computational information such as witnesses in first-order logic or resource consumption in programs. In this thesis we present a general framework for enriching causal concurrent games models with annotations able to reflect these pieces of information. These annotations can be of various nature, in particular our enrichment is parametrised over any multi-sorted equational theory and can also reflect structure upon it such as a partial order.

In our model, annotations on strategies can be viewed as side-computations: the information they reflect is modified throughout interactions but does not affect the general flow of control. From a semantics point of view, this construction is motivated by two semantic problems from both logic and programming languages :

1. On the logic side, our annotated games model specialised to first-order terms enables us to give a novel interpretation of first-order classical proofs as concurrent strategies carrying first-order witnesses. In particular, this answer the question of giving a compositional version to Herbrand's theorem while avoiding the usual proof sequentialization of other denotational approaches.
2. On the programming language side, annotations on games offer intrinsic quantitative models. We show that those can be used to provide denotational semantics for resource consumption analysis of concurrent higher order programming language with shared memory.

These enrichments, strongly connected to the causal structure of concurrent games, give an argument in favor of a causal and event-base meaning of computations.

Merci,

à toutes celles et ceux qui n'oublient pas de sourire
à ceux qui doutent
à celles qui mettent le doute
à ceux qui partagent sans compter
à celles qui ne calculent pas
à toutes celles et ceux qui résistent

Merci,

à celles qui m'instruisent
à ceux qui m'inspirent
à celles qui m'écoutent
à ceux qui me font rire
à mes amours, à mes amis
à Amandine, Françoise et Pierre et au reste de ma famille

Merci,

à la terre qui nous supporte
au temps qui passe
à cette chienne de vie
à l'amour

Merci enfin et surtout,

à Pierre pour sa patience, sa droiture et l'envie de partager sa science
à Olivier pour ses conseils avisés
à Glynn et Martin pour leur bienveillance
à mes rapporteurs, Dan Ghica et Dale Miller pour leurs précieuses
relectures
à Claudia Faggian et Delia Kesner pour avoir également accepté
de faire partie de mon jury

à la communauté pour son intérêt
aux membres de Plume, passés et présents
à la dream team CG : Simon, Hugo et Marc
aux thésard(e)s du LIP et à ceux de Cambridge
à mes étudiants qui m'en ont fait apprendre
à l'équipe MALIP pour son soutien précieux

Contents

Introduction (Français)	1
Introduction (English)	11
1 Preliminaries on plain concurrent games	19
1.1 Category of event structures	19
1.2 Category of concurrent games and strategies	25
1.2.1 Games and strategies as event structures with polarities	25
1.2.2 Interaction of strategies.	28
1.2.3 Composition of strategies	31
1.3 Compact closed structure	37
I Annotated strategies	43
Introduction	45
2 Concurrent games with annotations	47
2.1 Σ -strategies	47
2.1.1 Preliminaries on terms	47
2.1.2 Σ -strategies	51
2.2 \mathbb{T} -strategies	57
2.2.1 \mathbb{T} -algebras	57
2.2.2 A category of \mathbb{T} -strategies	60
2.3 Examples of \mathbb{T} -strategies	64
2.3.1 $\text{CG}_{\mathbb{R}}$	64
2.3.2 CG_{Set}	66
2.3.3 \mathcal{C} -CG	68
3 Categorical structure of \mathbb{T}-CG	71
3.1 A category of Σ -event structures	71
3.1.1 Σ -event structures	72
3.1.2 Pullbacks	74
3.1.3 Partial morphisms and projections	77
3.2 Compact closed structure of Σ -CG	82
3.2.1 From Σ -morphisms to Σ -strategies	82
3.2.2 Σ -CG: a compact closed category	85

CONTENTS

3.3	Observations and remarks	89
4	Simpler models	93
4.1	Games and rigid strategies	93
4.1.1	The compact closed structure of Strat	94
4.1.2	From CG to Strat	101
4.2	Elementary games and strategies	105
4.2.1	The compact closed structure of Det	105
4.2.2	Relation with Strat	111
4.3	Putting annotations back	112
4.3.1	T-augmentations	112
4.3.2	T-Rig	116
4.3.3	T-Det	119
II	Term-strategies for Herbrand's theorem	123
	Introduction	125
5	From Herbrand proofs to winning strategies	129
5.1	Preliminaries on LK_1	129
5.1.1	First order classical formulas	129
5.1.2	Classical sequent calculus	132
5.2	Herbrand's theorem	138
5.2.1	Herbrand proofs	139
5.2.2	Expansion trees.	140
5.3	Games for Herbrand's theorem	142
5.3.1	Expansion Trees as Winning Σ -Strategies	142
5.3.2	A (biased) interpretation of formulas	146
6	A model for first order MLL	151
6.1	Winning Σ -strategies	151
6.1.1	*-autonomous structure	152
6.1.2	Interpretation of MLL	157
6.2	Interpretation of MLL_1	158
6.2.1	A fibred model of \mathcal{V} -MLL	160
6.2.2	Linear quantifiers	163
7	An interpretation of LK_1	171
7.1	Interpretation of LK_1 proofs	171
7.1.1	Exponential functors	173
7.1.2	Contraction	175

CONTENTS

7.1.3	Interpretation of proofs	178
7.2	Compositional Herbrand's theorem	178
7.3	Discussion on the computational content of $\llbracket - \rrbracket_{\mathcal{V}}$	183
7.3.1	Cut reductions	183
7.3.2	Non-finiteness of the interpretation	186
III Resource tracking concurrent games		193
	Introduction	195
8	Semantics of \mathcal{R}-IPA	199
8.1	Operational semantics of \mathcal{R} -IPA	199
8.1.1	Affine IPA	200
8.1.2	Cost semantics and \mathcal{R} -IPA	204
8.1.3	Non-interleaving operational semantics	207
8.2	\mathcal{R} -strategies for \mathcal{R} -IPA	210
8.2.1	Arenas and rigid \mathcal{R} -Strategies	211
8.2.2	Negative interpretation of \mathcal{R} -IPA	215
8.2.3	Pre-orders on \mathcal{R} -strategies	217
9	Soundness and Adequacy	221
9.1	Soundness for \mathcal{R} -IPA	221
9.1.1	Interpretation of memory states	222
9.1.2	Single step	226
9.1.3	Many steps	227
9.2	Adequacy for \mathbb{R}_+ -IPA	230
9.3	Improvement for \mathbb{R}_+ -IPA	235
9.3.1	Semantics of Improvement	236
9.3.2	Examples of improvement	238
9.3.3	Further improvements.	240

Introduction (Français)

La sémantique des jeux est une sémantique dénotationnelle centrée sur l'interaction : les preuves et les programmes y sont représentés par des stratégies modélisant, par le flot d'exécution, leur manière de réagir à leur environnement. Malgré cette présentation intensionnelle, les sémantiques de jeux ne suffisent pas à capturer certaines informations calculatoires annexes au flot d'exécution telles que, par exemple, la production de témoins en logique du premier ordre ou la consommation de ressources dans les langages de programmation.

Dans cette thèse, nous proposons une construction générale permettant d'enrichir le modèle des jeux concurrent à base de structures d'événements avec des annotations afin de garder trace de ces informations. Nous appliquons ensuite cette construction sur les deux exemples précédemment mentionnés.

Sémantique de jeux

Les sémantiques de jeux sont des sémantiques *compositionnelles* (dénotationnelles) pour les preuves et les programmes. Très *intensionnelles*, elles représentent le calcul comme une *interaction* entre un *joueur* (P) et un *opposant* (O) : le programme et son environnement.

Sémantique des jeux pour les preuves

La sémantique des jeux pour les preuves [Coq95, Bla92, AJ94, HDP93, BDER97, Mel05, Lau10] est héritière des travaux de Lorenzen [Lor60] qui voit les formules logiques comme des jeux dont les règles caractérisent la manière dont deux philosophes (ou joueurs) peuvent débattre de la véracité d'une formule (l'un cherchant à prouver la formule, l'autre à la l'infirmier). Cette interprétation s'étend naturellement aux preuves qui sont alors vues comme des *stratégies*, c'est à dire des manières de débattre (*interagir*) avec un Opposant. Toutes les stratégies ne sont pas bonnes pour convaincre Opposant, on se restreint donc aux stratégies *gagnantes* c'est à dire aux stratégies qui permettent de *gagner* le débat (aussi appelée la *partie*) à chaque fois. Les *conditions de gain* d'une partie sont définies à l'avance : elles sont spécifiées dans les règles du jeu.

Dans les sémantiques de jeux pour la logique, une formule est interprétée par un jeu. Par exemple, une formule dont le connecteur principal est le connecteur propositionnel \wedge sera interprétée comme un jeu pour lequel Opposant commence et choisit l'une des deux sous-formules (la composante droite ou la composante gauche du connecteur) sur laquelle il veut continuer le débat. Ce choix correspond à un *coup* ; une fois le coup joué, le jeu continue sur la sous-formule choisie. Prenons pour autre exemple une formule commençant par un quantificateur existentiel \exists : cette fois, le quantificateur correspondra à un coup Joueur et Joueur devra fournir un *témoin* permettant d'instancier la formule sous le quantificateur. Une fois le coup joué le jeu continuera sur cette sous-formule.

Un des points clé des modèles de jeux dans l'interprétation de preuve est leur structure catégorique : les modèles de jeux permettent de *composer* les stratégies. En terme de preuves, cela signifie que ces modèles sont capables d'interpréter la règle de coupure : la formule coupée est vue comme un jeu commun sur lequel les deux stratégies peuvent interagir avec des intérêts divergents (jeux duaux). Cette interaction est interne à la composition des deux stratégies et est donc caché aux yeux d'un environnement externe qui chercherait à invalider les formules restantes.

L'exploration des sémantiques de jeux pour la logique à mener à divers modèles de preuves préservant l'élimination des coupures notamment en logique linéaire propositionnelle ou intuitionniste, ou pour des versions polarisées de la logique classique, parfois étendues au premier ordre [AJ94, HDP93, BDER97, Mel05, Lau10].

À retenir : Les sémantiques de jeux pour les systèmes logiques interprètent les formules par des jeux et les preuves par des stratégies gagnantes. Ces modèles de jeux sont compositionnels, ils permettent d'interpréter l'élimination des coupures des systèmes considérés, et induisent donc une notion de calcul.

Sémantique des jeux pour les programmes

Généralisant les idées issues des sémantiques de jeux pour les systèmes de preuves, les sémantiques de jeux pour les langages de programmations interprètent les types comme des jeux et les programmes comme des stratégies sur ces jeux. Cette vision suit l'idée qu'un programme est contraint par son type et qu'il décrit le comportement d'un système par rapport à son environnement.

Oubliant le plus souvent l'idée de gain, les modèles de jeux utilisés pour

dénotés des programmes restent des modèles de jeux à deux joueurs (le Programme et son environnement) où les stratégies peuvent être composées par interaction cachée. Dans ces modèles c’est donc la représentation des interaction (parties) et du *contrôle* du calcul induit qui prime. En particulier, les coups d’un jeu représentent les *événements calculatoires observables* entre un programme et son environnement et les règles du jeu en régulent (partiellement) l’ordre. Chaque coup joué représente un passage de contrôle. En plus du jeu, ce passage de contrôle peut également être régulé par des contraintes générales supplémentaires, permettant de restreindre les effets du calcul représenté.

Comparativement à d’autres modèles dénotationnels tels que les domaines [Sco82], les modèles de jeux offrent une structure plus riche pour l’interprétation des types et des programmes. De part leur représentation du flot de contrôle ils sont très proches des sémantiques opérationnelles (c’est à dire des sémantiques représentant le calcul par des systèmes de transition), sans pour autant reposer sur la syntaxe des langages considérés. Pour cette raison on dit parfois que les sémantiques de jeux sont des sémantiques dénotationnelles “intensionnelles”, par opposition aux sémantiques dénotationnelles précédentes plus *extensionnelles*.

De part leur représentation fine du flot d’exécution, les modèles de jeux se sont illustrés en sémantique des programmes, notamment en apportant une réponse au problème fondamental de *pleine adéquation* (*full abstraction*) pour PCF (un langage d’ordre supérieur avec point-fixe en appel par nom). Ces réponses, indépendamment proposées par Hyland et Ong [HO00] et Abramsky, Jagadeesan and Malacaria [AJM00], ont ouvert la voie à une variété d’autres modèles pleinement adéquats pour des langages combinant d’autres primitives de calcul ou effets calculatoires tels que l’appel par valeur [HY97], la mémoire (Idealised Algol) [AM96], les opérateurs de contrôle [Lai97], les références [AHM98], les exceptions [Lai01], le non-déterminisme [HM99], la concurrence [Lai06] ou les calculs probabiliste [DH02].

À retenir : La sémantique des jeux pour les langages de programmations capture la notion de calcul par celle d’interaction : un programme correspond à un ensemble de parties (*i.e.* une stratégie) sur un jeu représentant son type ; chaque partie décrit une interaction possible du programme avec son environnement. Les sémantiques de jeux sont modulaires : en faisant varier les structures sous-jacentes à la description d’un jeux et d’une partie, les modèles de jeux donnent une caractérisation à des primitives de calcul variées.

Modèle de jeux concurrents Les modèles de jeux fondateurs en sémantique pour les langages de programmation sont des modèles de jeux *séquentiels* [HO00, AJM00] : lors d’une partie, les coups Joueur et Opposant alternent, signifiant qu’un joueur doit attendre que son adversaire ait joué avant de pouvoir jouer de nouveau, forçant ainsi une interaction linéaire/séquentielle.

Pour interpréter des programmes concurrents, une première modification aux modèles de jeux séquentiels a été donnée par Ghica et Murawski [GM08]. Leur modèle de jeux est dit par *entrelacements* car dans ce modèle, la contrainte d’alternance sur les interactions est supprimée : une interaction peut donc être vu comme un entrelacement d’interactions séquentielles, un même joueur pouvant jouer plusieurs fois de suite (tant que le jeu le permet) sans attendre la réponse de son opposant. Ce comportement correspond par exemple au fait qu’un programme peut lancer plusieurs calculs en parallèle sans attendre que l’un termine avant de lancer le suivant. Ce modèle est pleinement adéquat pour la “may”-équivalence.

Dans le modèle par entrelacement, toutes les chronologies possibles d’un calcul parallèle sont représentées. Pour éviter cette énumération fastidieuse mais aussi pour donner une sémantique plus centrée sur le flot de contrôle que sur la chronologie des événements calculatoires, un second modèle de jeux pour la concurrence, appelé modèle de jeux causaux ou encore modèle *vraiment concurrent* a été introduit par Abramsky and Melliès [AM99] puis étendu plus tard par Melliès et Mimram [Mel05, MM07] et Faggian et Piccolo [FP09]. Dans ce modèle, les interactions ne sont plus représentées par des séquences de coups (ordres totaux) mais par des ordres partiels capturant les relations de cause à effet entre les coups d’une partie. Dans ce modèle la sémantique d’un programme est donc abstraite du choix de l’ordonnancement à l’exécution.

C’est dans la lignée de ces modèles de jeux causaux que s’inscrit le modèle de jeux présenté dans cette thèse. En particulier notre modèle est un enrichissement du modèle de jeux sur les structures d’événements développé par Clairambault, Castellan et Winskel [CCW15, CC16, CCHW18] à partir des travaux de Rideau and Winskel [RW11, CCRW17]. Ce modèle de jeux a été activement enrichi ces dernières années, notamment afin de prendre en compte les calculs (concurrents) probabilistes [CCPW18, CP19] ou quantiques [CdVW19].

À retenir : Les modèles de jeux concurrents sont des modèles causaux : les parties sont décrites par des ordres partiels de coups plutôt que par des séquences de coups. En sémantique dénotationnelle, cela donne une vision des programmes très proches de leur flot de contrôle.

Motivations et résultats

À une large échelle, cette thèse participe à l’exploration de l’expressivité des sémantiques dénotationnelles centrées sur le flot d’exécution (vision causale et événementielle du calcul) en prenant pour point de départ le modèle des jeux concurrent sur les structures d’événements [CCRW17] (ci-après simplement nommé modèle de jeux concurrent).

Dans cette thèse, nous répondons en fait à deux problèmes de sémantiques pour les preuves et les programmes pour lesquelles la nature causale des jeux concurrents semble appropriée :

1. **Rendre le théorème de Herbrand compositionnel:** Le théorème de Herbrand est un théorème fondamental en logique classique du premier ordre qui relie la validité d’une formule existentielle à celle d’une disjonction finie correspondant à cette même formule, instanciée par un nombre fini de termes clos appelés *témoins de Herbrand* de la formule. Étant donnée une preuve classique pour une formule valide, il est possible d’utiliser cette preuve pour calculer un ensemble de témoins de Herbrand pour cette formule. Contrairement aux preuves, les témoins de Herbrand ne peuvent cependant *pas être composés*. Une question d’intérêt en théorie de la preuve est de comprendre quelles structures mathématiques plus simples que les preuves peuvent capturer ces témoins, tout en restant compositionnelles [Hei10, McK13, HW13].

Les modèles dénotationnels pour les systèmes de preuves sont par essence compositionnels. Un modèle dénotationnel pour les preuves de la logique classique serait donc un bon candidat pour décrire de telles structures. Cependant, très peu de sémantiques dénotationnelles ont été développées pour la logique classique [FP07]. Cette défection est liée au fait que la théorie équationnelle de l’élimination des coupures du système de preuves classique est dégénérée et que l’on préfère donc représenter des systèmes de preuves mieux formés et logiquement équivalents tels que les système de preuves *polarisés* [Par92, Lau10].

Pourtant, dans les travaux les plus récents autour du théorème de Herbrand, la structure syntaxique des arbres expansés (*expansion trees*) qui généralise les témoins de Herbrand est souvent introduite au travers une métaphore de jeux permettant de refléter la structure causale intrinsèque de ces témoins [Mil87, Hei10, HW13]. Forts de ces constats, nous avons enrichi le modèle de jeux concurrent en augmentant les stratégies de termes du premier ordre permettant d’interpréter les preuves de la logique classique du premier ordre. En conséquence nous avons obtenu une version compositionnelle du théorème de Herbrand.

2. **Avoir un modèle dénotationnel capable de rendre compte de l'utilisation de ressources pour des calculs concurrents:** La sémantique dénotationnelle pour les programmes trouve sa source dans l'étude de l'équivalence entre programme notamment pour donner une bonne fondation à des manipulations de programmes telles que la compilation. Habituellement on s'intéresse donc à des modèles dénotationnels *qualitatifs*, c'est à dire cherchant à capturer des propriétés sur les programmes qui sont stables par réductions, telles que la terminaison ou le résultat d'un calcul.

S'abstrayant des aspects les plus opérationnels du calcul, peu de modèles dénotationnels permettent d'en représenter des propriétés *quantitatives* tels que leur efficacités en terme d'énergie, de temps ou de consommation de ressources. Pourtant ces propriétés sont au cœur des recherches en optimisations, une branche non négligeable de l'informatique en ce qui concerne la transformation de programme. Il semble donc important de proposer une sémantique pour ces transformations.

L'un des premiers (et rares) travaux menés dans cette direction [Ghi05, LMMP13], est issu de la sémantique des jeux et donne une interprétation pour l'amélioration du temps d'exécution de programmes concurrents. Ce modèle, appelé modèle de *jeux à jetons* (*slot games* [Ghi05]), est un enrichissement du modèle de jeux par entrelacements [GM08], il ne peut donc pas rendre compte du temps d'exécution de deux threads réellement exécutés en parallèle ; dans ce modèle, tout se passe comme si le programme multi-thread était exécuté sur une machine à un seul cœur. Pour rendre compte du temps d'exécution d'un calcul réellement parallèle, nous avons donc cherché à enrichir le modèle de jeux concurrent en augmentant les stratégies de fonctions permettant de représenter l'utilisation de certaines de ressources telles que le temps. De cette construction nous avons tiré une interprétation correcte et adéquate du langage \mathbb{R} -IPA, un langage concurrent d'ordre supérieur avec mémoire partagée dont la sémantique opérationnelle donne une représentation explicite du temps d'exécution.

Donner un cadre général pour enrichir les jeux concurrents d'annotations De part sa représentation du flot de contrôle le modèle de jeux concurrent nous a offert le socle désiré pour construire les sémantiques présentées ci-dessus. Cependant, dans les deux cas, le modèle a du être enrichi pour refléter des aspects intéressants du calcul qui n'ont pas d'impact sur le flot de contrôle et ne sont donc pas visibles dans le modèle tels que la production de témoins en logique du premier ordre ou l'utilisation de ressources

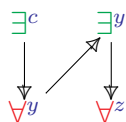


Figure 1: Exemple de stratégie annotée issue de la partie II

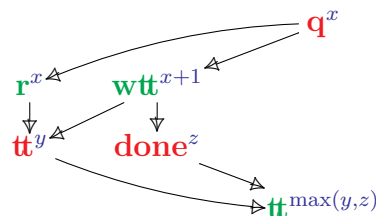


Figure 2: Exemple de stratégie annotée issue de la partie III

des programmes concurrents. Dans les deux cas, ces enrichissements revenaient à donner plus de sens au fait de jouer un coup : dans le premier cas, jouer un coup \exists va de paire avec le fait de donner un témoin/terme du premier ordre ; dans le second cas, jouer un coup c'est aussi garder trace du temps d'exécution nécessaire pour jouer ce coup.

Nous avons montré que ces enrichissements se généralisent dans un nouveau modèle de jeux concurrent avec *annotations* dans une *théorie équationnelle multi-sortée* : dans ce modèle chaque coup dans un jeu est associé à une sorte telle que ce coup ne peut être joué au cours d'une partie que si son joueur fournit en même temps un terme clos (une valeur) de cette sorte. La relation de cause à effet entre les coups est ici importante car la valeur jouée peut être utilisée par le joueur adverse pour produire ses propres valeurs (dans le cas où les coups correspondant dépendent du coup joué). La valeur d'un coup peut donc varier en fonction des interactions bien qu'elle n'a pas d'influence sûr ces dernières : les aspects "quantitatifs" du modèle enrichi n'influent pas sur les aspects "qualitatifs" du modèles de départ. En particulier le modèle enrichi préserve la structure compacte fermée du modèle de jeux concurrent sans annotation.

Dans cette thèse nous soulevons également la question d'un modèle de jeux annoté dans lequel les annotations auraient un impact sur les interactions mais conserveraient tout de même une structure catégorique compacte fermée. Cette question n'est cependant pas pleinement développée car elle dépasse le cadre des problèmes sémantiques traités dans cette thèse.

Résumé des chapitres.

Préliminaires

Chapitre 1 : donne un rappel des constructions du modèle des

jeux concurrents sans annotations telle que décrites dans [CCRW17]. On insiste particulièrement sur la notion de composition entre stratégies concurrentes et rappelle les méthodes de preuves utilisées pour montrer la structure catégorique compacte fermée du modèle engendré.

Partie I : Stratégie annotées

Chapitre 2 : présente la construction générale pour notre modèle de jeux annoté. Ce modèle, appelé \mathbb{T} -CG, est paramétré par une théorie (in-)équationnelle multi-sortée \mathbb{T} . Cette construction générale est présentée de manière introductive au travers différents exemples en particulier pour des annotations de termes et de fonctions.

Chapitre 3 : détaille la preuve de la structure catégorique compacte fermée de \mathbb{T} -CG. Cette preuve s'inspire largement des méthodes de preuves rappelées dans le chapitre 1 mais introduit une nouvelle catégorie de structure d'événements annotées. On discute brièvement de l'utilisation de cette catégorie pour construire un nouveau modèle de jeux concurrent annoté dans lequel les annotations peuvent perturber les interactions.

Chapitre 4: conclut la partie I en présentant deux versions simplifiées du modèle annoté qui seront effectivement utilisées dans les parties II et III. Ces simplifications sont en fait des enrichissements des versions simplifiées du modèle de jeux concurrent sans annotations, connus comme modèle de *jeux et stratégies élémentaires* (*elementary games and strategies*) et modèle de jeux à *stratégies rigides* (*concurrent games and rigid strategies*) dont nous rappelons également les constructions.

Partie II: Stratégies à termes pour le théorème de Herbrand

Chapitre 5 : donne une introduction brève à la logique classique du premier ordre et au théorème de Herbrand dans sa version moderne c'est à dire exprimée en terme d'arbres expansés. On y donne également une interprétation de ces arbres en terme de stratégies élémentaires annotées par des termes. Cette interprétation nécessite l'ajout de *conditions de gains* au modèle de jeux décrit dans le chapitre 4.

Chapitre 6 et Chapitre 7 : présente l'interprétation des preuves de la logique classique du premier ordre (LK_1) en terme de stratégies gagnantes annotées. On se concentre d'abord sur l'interprétation de MLL_1 (le fragment

multiplicatif de la logique linéaire du premier ordre) puis on interprète les règles de contraction et d'affaiblissement, complétant ainsi l'interprétation de LK_1 . Cela nous permet de conclure sur notre version compositionnelle du théorème de Herbrand et d'analyser quel comportement calculatoire des preuves de LK_1 se trouve reflété dans notre modèle.

Partie III: Jeux concurrents pour l'analyse de ressources

Chapitre 8 : présente le langage \mathcal{R} -IPA, une version du langage \mathbb{R} -IPA dont la sémantique opérationnelle est paramétrée par le type de ressources à analyser (le temps dans le cas de \mathbb{R} -IPA). On rappelle brièvement l'interprétation de ce langage dans le modèle de jeux à jetons pour le mettre en regard avec la sémantique opérationnelle *vraiment parallèle* que l'on se propose d'étudier. On donne ensuite l'interprétation de cette version vraiment parallèle du langage dans le modèle de jeux annoté par des fonctions, en prenant pour modèle la version simplifiée présentée dans le chapitre 4 et restreinte aux jeux négatifs et bien filés.

Chapitre 9 : présente finalement la preuve de correction du modèle pour \mathcal{R} -IPA. On y montre aussi que ce modèle est adéquat dans le cas du temps. Ceci n'est pas le cas en général car la sémantique opérationnelle du langage dans le cas général n'est pas aussi fine que sa sémantique dénotationnelle. On conclut cette partie en proposant une sémantique pour *l'amélioration* des programmes concurrents de \mathbb{R} -IPA.

Introduction (English)

This thesis presents a general framework for enriching *causal concurrent games* with *annotations*. From a semantics point of view, this construction is motivated by problems from both logic and programming languages.

On the logic side, an annotated games model specialised to first-order terms yields a setting in which to interpret *first-order classical proofs* and give a *compositional version* to the foundational Herbrand's theorem. On the programming language side, annotations on games offer intrinsic *quantitative* models that can be used to construct a denotational semantics for resource consumption of higher order programs with state and concurrency.

Game semantics

Over the rich field of denotational semantics for proofs and programs, game semantics is usually considered as one of the most *intensional* semantics: game models provide abstract frameworks to capture the meaning of computations as *interactions* with an external environment, an Opponent.

Game semantics for proofs

Game semantics for proofs extends the seminal idea of Lorenzen [Lor60] according to which the *meaning* of a *formula* can be understood as a *game ruling* the way in which two *players*, a *Proponent* P and an *Opponent* O , can *dialogue* (or *play*, or *interact*) in order to respectively prove or disprove the formula. In a game, *turns* are represented by *moves* – attributed either to Proponent or to Opponent – and, in order to determine who from Proponent or Opponent makes the better point, a game also has *winning conditions* that specify the winner at the end of an interaction/play.

For example, if the main connective of a formula is \wedge , it is Opponent's turn to speak (*play a move*) to choose which of the two sub-formulas is then going to be discussed. Ultimately, the winner of that game will be the winner of the chosen formula. As another example, in first order logic, if the formula is an *existential* formula (that is a formula starting with a \exists quantifier) then it is Proponent's turn to play and she has to provide a *witness* (a first-order term) before the two players keeps playing over the formula instantiated with this witness.

As incrementally shown by the game semantics community [Coq95, Bla92, AJ94, HDP93, BDER97, Mel05, Lau10], this game semantics of formulas, naturally extend to *proofs*, viewing the latter as *strategies for P*, that is, as descriptions of how Proponent should react to Opponent during a play in order to win. Since a proof for a formula cannot be refuted, game models of proofs are restricted to *winning strategies*, that are strategies winning against every Opponent.

A foundational aspect of game models being models of proofs, is that they enjoy the categorical properties of the corresponding proof system, in particular they have a notion of *composition*, realised via the interaction between two strategies on *dual* games, that are, games interpreting a formula and its negation.

To sum up: in games semantics for proofs, games models are made of games (interpreting formulas) and winning strategies (interpreting proofs), and, as they interpret cuts, they also support a notion of composition/computation.

Game semantics for programs

Although coming from logics, game semantics is better known for its success in *denotational semantics*, that is, the study of the mathematical meanings of programming languages

Following intuitions from the *Curry-Howard correspondence* that views formulas as particular types and proofs as particular programs, in game semantics for programming languages types are interpreted as games and programs as strategies. In particular, this reflects the idea that types “rule” the way in which programs run and that programs are syntactic description of how a system should react to its environment.

In general and contrary to games for logics, games for programming languages do not have winning conditions. Here the focus is kept on interactions and how they are *controlled*. In particular games interpreting types have moves corresponding to the *computational events* of these types and are organised accordingly.

As for proofs, games for types are *two-player games*. Their moves are distributed over Player and Opponent according to whether the corresponding computational event gives control to the Program or to its environment. With more structure than earlier interpretation of types as *domains* [Sco82] game models provide a richer space to interpret programs and in particular to reflect their control flow.

For these reasons, games models are often viewed as lying in between abstract denotational semantics and *operational semantics* (*i.e.* the syntactic formalization of their execution via transition systems): they enjoy the compositional structure of the first one (making them suitable to interpret contexts and open terms) while being rich enough to give a fine grained view of the control flow of the second ones.

This mixed nature of games models is illustrated in the foundational game semantics answers to the *full abstraction problem* for PCF (a call-by-name higher order programming language with fixpoint operators) independently provided Hyland and Ong [HO00], and by Abramsky, Jagadeesan and Malacaria [AJM00]. As surveyed in [Pan17], these game semantics have since then been successfully enriched and/or generalised to provide denotations for various other computational features such as call-by-value [HY97] state (Idealised Algol) [AM96], control [Lai97], references [AHM98], exceptions [Lai01], non-determinism [HM99], concurrent and mobile processes [Lai06] and probability [DH02].

To sum up: in games semantics for programming languages, the meaning of programs is given as strategies on games representing types. These strategies are described by *plays* which represent the various way in which a program may *interact* with its environment, constrained by the rules and moves/computational events of its type. Games models are modular frameworks: by varying the way in which they represent or constrain plays they offer semantics to different kinds of computational features.

Concurrent game models

In the original sequential game semantics [HO00, AJM00], interaction between programs are represented as *sequences* of moves alternating between Player and Opponent (the control flow of an interaction is then sequential as each player must wait that for its opponent to reply before being allowed to perform another move). In order to capture concurrency in programming languages Ghica and Murawski described a model of games in which this alternating condition is removed [GM08]. This interpretation of concurrency yields an *interleaving-based* fully abstract model of games for concurrent programming languages with respect to may equivalence. Here, that the model is interleaving-based means that concurrency is captured by enumerating all the possible chronological orderings between computational events.

To avoid such an enumeration and to remain more faithful to the causal (“control-flow-based”) view of computation, an other family of games models has been developed: the family of *truly concurrent* games models. Rather than defining concurrency via every possible linear ordering of moves, these

models chose to represent it via *partial orders* putting instead the emphasis on the (in)dependencies between moves. In terms of games semantics for concurrent programming languages this abstracts away all the choices of the scheduler that are irrelevant from a computational point of view, namely, the reorderings of independent events.

Causal games semantics was pioneered by Abramsky and Melliès [AM99] and pushed forward by Melliès [Mel05] with Mimram [MM07], and by Fag-gian and Piccolo [FP09]. In the past few ten years, they have been actively developed by Clairambault, Castellan and Winskel [CCW15, CC16, CCHW18] with Paquet [CCPW18, CP19], and de Visme [CdVW19]; originally prompted by the introduction of a more general framework by Rideau and Winskel and based on event structures [RW11, CCRW17]. It is upon this last model, from now on referred to as *the concurrent games model*, that the work presented in this thesis is built.

To sum up: in concurrent games model the description of interactions is given through (partial) orders of moves rather than sequences of moves. This put the emphasize on the causal relation of computational events rather than on their chronological order. In the case of concurrent programming languages this provides a more compact representation of programs than interleaving-based models.

Motivations and results

Generally speaking, this thesis is motivated by the exploration of the expressiveness of a causal and event-based representation of computations using concurrent games.

More precisely, in this thesis we address two semantic problems (introduce with more details at the beginning of parts II and III) whose causal nature prompts to an interpretation in the concurrent games model:

1. **A compositional interpretation of Herbrand’s theorem:** Herbrand’s theorem is a foundational theorem in first-order classical logics that reduces validity of an existential formula to that of the disjunction of its instantiation with finitely many closed terms called the *Herbrand witnesses* of the formula. Although extracted from proofs, these witnesses cannot be composed: in general, given witnesses for $\vdash A$ and $\vdash A \implies B$ there is no direct way to deduce witnesses for $\vdash B$ [Koh99]. Understanding how the structure of these witnesses can be elaborated to allow composition (while preserving their original compactness) has thus become a question of interest in proof theory, in particular as a way to design alternative proof formalisms [Hei10, McK13, HW13].

Herbrand’s theorem is also foundational in that its witnesses give a computational content to cut-free first order classical proofs. On the other hand, the computational content of proofs is a central question in the Curry-Howard community, and denotational models are intrinsically compositional. Yet, as the equational theory of cut elimination of classical logic is degenerated, only few studies have been carried on to give an interpretation to the classical sequent calculus [FP07]; the preference is given to better behaved systems, equivalent in terms of provability, such as polarized versions of classical logic [Par92, Lau10].

In modern syntactic approaches to Herbrand’s theorem however, Herbrand’s witnesses are represented as certain trees (called *expansion trees*) and given an intuitive game semantics to describe some of the causal structure intrinsic to these witnesses [Mil87, Hei10, HW13]. Based on these works, we constructed a model of concurrent games enriched with first-order annotations on events, suitable to embed expansion trees and interpret first-order classical proofs. Through semantics, this yields a compositional version of Herbrand’s theorem.

2. **A denotational framework for parallel resource consumption:** Denotational semantics for programming languages are usually motivated by problems where understanding the notion of program equivalence is crucial as *e.g.* in program compilation. For this reason, denotational models for programming languages are often qualitative: they capture properties of computation that are invariant under reduction and ignore the others, such as efficiency of programs in terms of time, power or other resource consumption, often referred to as *quantitative properties*.

Yet, in the field of program optimisation, understanding the notion of program *improvements* is crucial and needs semantical insight. It is thus interesting to seek for *quantitative* denotational models of computation. Few studies have been carried along this line [Ghi05, LMMP13], one primary work however comes from games semantics and provides a semantical interpretation of concurrent program improvements with respect to time consumption [Ghi05]. This model, called slot games, is an enrichment of the interleaving-based model of [GM08] and as such does not account for a truly parallel way of consuming resources (it is as if the multi-threaded programs were executed on a one core machine instead of a multi-core one). It was thus natural to seek for an enrichment of the concurrent games model that would provide a truly

concurrent semantics for time consumption. So we did: endowing concurrent strategies with functional annotations, we provide a sound and adequate games semantics for \mathbb{R}_+ -IPA, a higher order concurrent programming language with state, whose operational semantics allows for parallel time consumption.

A concurrent games model with general annotations In both of the works presented above, the original concurrent games model was close to but not completely fit to reflect the intended meaning upon the proofs and programs under study. In a sense, the model was lacking *meanings on moves/events*. More precisely, in the case of first order-proofs, \exists -moves were lacking first-order witnesses, and, in the case of time analysis, computational events were lacking a record of the time they would take.

To add meaning to events, we enriched the setting of concurrent games with annotations: on games, moves are given a “kind”, such that, when playing a move of their own, P and O must also provide a value of that kind. This value might in turn be used by the other player to compute values for its own next moves (next being understood in the causal sense). This blends the meaning of events into the more general structure of plays and interactions.

Common to both models, we subsumed the idea of having annotations on events in a general framework that allows to enrich concurrent games models with annotations from any *multi-sorted inequational theory*. This enrichment faithfully preserves the categorical structure of the underlying model in the sense that removing annotations goes back to the plain concurrent games model. In particular, this means that annotations on events do not have impact on the “qualitative meaning” of strategies (their control flow) although they are themselves sensitive to variation that would otherwise be abstracted away/hidden by the model.

As an open-ended perspective for future work, we also intuit a more general model of annotated concurrent games in which annotations on events (or quantitative aspects of the computations) may impact the way in which players/strategies interact.

Outline of this thesis.

We give here a short summary of the various chapters composing this thesis.

Preliminaries

Chapter 1 recalls the constructions of the plain concurrent games model based on event structures as presented in [CCRW17]. In particular

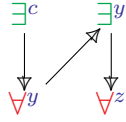


Figure 3: A strategy from part II

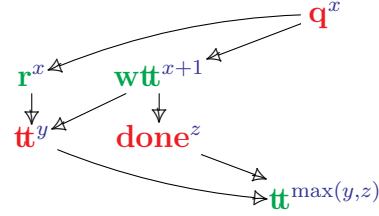


Figure 4: A strategy from part III

it introduces the notion of concurrent strategies and interactions upon them. In this chapter, we also remind the proof methodology to prove the categorical structure of the resulting model CG; a compact closed category.

Part I: Annotated strategies

Chapter 2 presents our general construction for concurrent games with annotations on events. It defines $\mathbb{T}\text{-CG}$, the parametric concurrent games model with annotations as terms from an *(in)equational theory* \mathbb{T} . This general construction is provided along with special cases of annotations, in particular with first-order terms and functions.

Chapter 3 provides a detailed proof of the categorical structure of $\mathbb{T}\text{-CG}$ – a compact closed category. The focus is actually put on $\Sigma\text{-CG}$, the annotated model specialised to terms whose categorical structure is equivalent to the one of $\mathbb{T}\text{-CG}$ after quotient. This proof is widely inspired from the proof of the categorical structure of CG reminded in chapter 1. Interestingly enough, it relies on the construction of a category of *annotated* event structures in which annotations *may interfere* with interaction.

Chapter 4 concludes on the construction of the annotated games model, by showing that, similarly to their plain counterpart, they can be simplified and provide an enrichment for the models of *elementary games and strategies* on the one hand, and *concurrent games and rigid strategies* on the other hand. These simple models are actually the ones in used in the applicative cases of part II and III.

Part II: Term-strategies for Herbrand’s theorem

Chapter 5 recalls first order logic and Herbrand’s theorem in its

more modern version as expansion trees. It introduces the exact game models in which expansion trees and proofs can be interpreted as strategies with first-order terms, in particular by adding *winning conditions* to the model introduced in chapter 4. This leads to a first reformulation of Herbrand's theorem.

Chapter 6 and **Chapter 7** describe the actual *interpretation of first order proofs* as winning strategies: in the first one we give the interpretation of *first-order multiplicative linear logic* (MLL_1); in the second one we add contraction and weakening to complete the interpretation of first order classical proofs (LK_1). From this we derive our compositional version of Herbrand's theorem and discuss some of the computational features of the LK sequent calculus reflected in our model.

Part III: Resource tracking concurrent games

Chapter 8 introduces the language \mathcal{R} -IPA, a general version of \mathbb{R}_+ -IPA mentioned above, for which the operational semantics is *parametrised* by the resource being analysed (time in the case of \mathbb{R}_+ -IPA). We sketch its interpretation in slot games and present its *parallel* operational semantics. We then present its interpretation as negative games and well-threaded strategies in the simplified model of rigid \mathcal{R} -strategies introduced in chapter 4.

Chapter 9 provides the proof of *soundness* of our model. It also shows *adequacy* for an operational semantics specialized to time, first noting that in general the parallel operational semantics is too coarse to fit the game semantics one. As a conclusion we give a semantic interpretation of the notion of *improvement* for concurrent programs of \mathcal{R} -IPA specialised to time.

Preliminaries on plain concurrent games

In this chapter we recall the constructions of the plain concurrent games model based on event structures as presented in [CCRW17]. In this model, games are event structures that describe how moves (events) connect with one another, and strategies are morphisms of event structures to their targeted games. This model is a *causal* game model, meaning that the total chronological ordering of events in a play is abstracted away into a more primitive notion of *causality*. This model also enjoys a notion of *consistency* that allows for describing non-determinism.

Section 1.1 first introduces the base category of event structures from which concurrent games and strategies are defined. In section 1.2 we make these definitions precise and show that concurrent strategies support a composition operation for which games have identities, thus defining a *category of concurrent games and strategies* called CG. Finally in section 1.3 we recall the constructions showing actually CG is a compact closed category. All results presented in this chapter can be found in [CCRW17] as well – with minor changes in proofs to better fit our needs. These results and their proof methodology are fundamental to the subsequent enrichment of concurrent strategies with annotations which is the core of this thesis and is developed in part I.

1.1 Category of event structures

This section introduces the basic category of event structures, \mathcal{E} , over which the concurrent games model à la Rideau and Winskel is built.

Event structures. Event structures were first introduced to reflect the behavior of concurrent and non-deterministic processes through the observation of event occurrences [Win86]. These occurrences are ruled by causality and consistency:

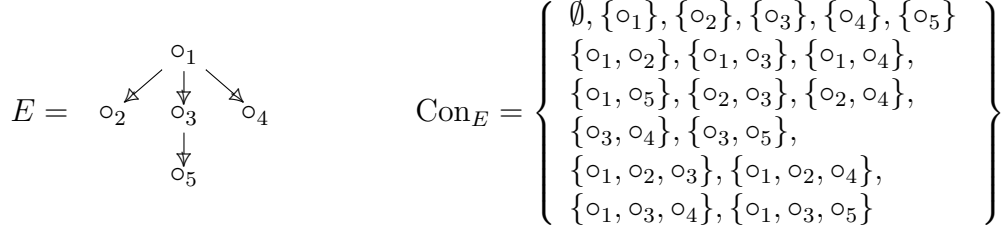
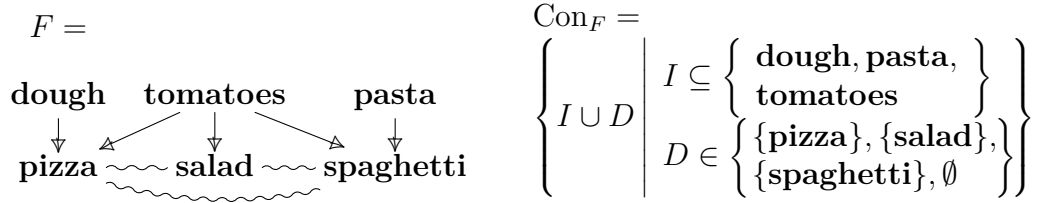
Figure 1.1: The graph E and Con_E form an event structure

Figure 1.2: Dinner making event structure

Definition 1.1 ([CCRW17]). An *event structure* is a tuple $(E, \leq_E, \text{Con}_E)$ where E is a set of *events*, \leq_E is a partial order on E called *causality* and Con_E is a non-empty set of finite subsets of E called *consistency*, such that:

- *finite history*: $\forall e \in E$ its *causal history*

$$[e] = \{e' \in E \mid e' \leq_E e\}$$

is finite;

- *reachability*: $\forall e \in E, \{e\} \in \text{Con}_E$;
- *down-closure*: $\forall X \in \text{Con}_E, \forall Y \subseteq X, Y \in \text{Con}_E$;
- \leq -*closure*: $\forall X \in \text{Con}_E, \forall e \in X, \forall e' \leq_E e, X \cup \{e'\} \in \text{Con}_E$.

We will often omit the subscripts in \leq_E, Con_E if they are obvious from the context, and use E both for the event structure and its underlying set of events.

Figure 1.1 and 1.2 are examples of event structures, their causal order is drawn from top to bottom (*i.e.* events at the top are minimal). We use them to comment on the definition.

Predecessors. For an event $e \in E$, the *strict* causality relation $<_E$ describes which events must occur before e can. These events are called the *predecessors* of e , defined by $e' \in E$ such that $e' \leq_E e$ and $e' \neq e$. Written $[e)$, the *set of predecessors of e* is finite and equal to $[e) = [e] - \{e\}$.

Since \leq_E is transitive and reflexive, it can be subsumed by the *immediate predecessor relation*,

$$e' \rightarrow_E e \quad \text{iff} \quad e' <_E e \quad \text{and} \quad \forall e'' \in E, \neg(e' <_E e'' <_E e)$$

that distinguishes predecessors of e with no other events in between. We have

Lemma 1.1. *For E an event structure, $\leq_E = (\rightarrow_E)^*$.*

The directed graph on the left of figure 1.1 can now be reread: it depicts a partial order where \circ_1 has no predecessor, and is the unique predecessor of events \circ_2, \circ_3 and \circ_4 , while \circ_5 has two predecessors: \circ_1 and \circ_3 .

Consistency. In definition 1.1, Con_E defines the set of *consistent subsets* of E . By condition *down-closure*, the set of consistent sets is closed under set-inclusion. The other axioms ensure that the history of every event is consistent. More generally, consistent sets are coherent with respect to \leq_E : if $X \subseteq E$ is consistent then its *down-closure* $[X]_E = \{e' \in E \mid \exists e \in X, e' \leq e \in X\} (= \bigcup_{e \in X} [e])$ is consistent.

Consistency is sometimes described using a *binary conflict relation*, $e \#_E e'$, that is an irreflexive and symmetric relation that is up-closed with respect to \leq_E (i.e. for every $e \#_E e'$ and $e' \leq_E e''$ then $e \#_E e''$). In that case, a subset $X \subseteq E$ is consistent if no event is in conflict with another. This is clearly closed under set-inclusion and up-closure of $\#_E$ with respect to \leq_E ensures that Con_E is also down-closed with respect to \leq_E . This characterisation of consistency is less general than the one in definition 1.1. For example consistency of E on figure 1.1 cannot be described using conflicts. Yet, using a conflict relation when possible has an advantage of conciseness as it can always be subsumed by a *minimal conflict relation* (\sim):

$$e \sim e' \quad \text{iff} \quad e \# e' \quad \text{and} \quad \forall e'' \in E, \begin{cases} e'' <_E e \Rightarrow \neg(e'' \# e') \\ e'' <_E e' \Rightarrow \neg(e'' \# e) \end{cases}$$

The consistency of F in figure 1.2 implies that only one dish can be cooked for dinner – maybe because there are not enough ingredients to make everything or maybe because the cook is tired. This consistency is simple enough to be described by a conflict relation whose minimal conflicts are depicted directly on the partial order F .

Binary conflict relations are expressive enough to capture the semantics of proofs and programs presented in Part II and III. For simplicity, in the rest of this thesis, examples will be based on event structures with binary conflicts, although the model will remain defined for general consistency.

Configurations Event structures characterise sets of configurations: for E an event structure, $x \subseteq E$ is a *configuration* of E if it is finite, down-closed and consistent. Configurations are sometimes viewed as partial orders by restriction of \leq_E ; we then write $\leq_x = \leq_E \cap x^2$. The set of configurations of E is denoted $\mathcal{C}(E)$. The empty configuration, \emptyset , or the causal histories, $[e]$, are examples of configurations. These latest configurations, with a unique top element, are called *prime* configurations.

Configurations are partially ordered by inclusion. In fact, this partial order can be subsumed by the *covering relation*

$$x \text{---} \subset y \quad \text{iff} \quad \exists e \in E, y - \{e\} = x$$

alternatively written $x \xrightarrow{e} \subset$ when one wants to put the emphasis on the *atomic extension* e . Starting from the empty configuration, configurations can be reached inductively as chains of atomic extension:

Lemma 1.2. *Let E be an event structure and $x, y \in \mathcal{C}(E)$ such that $x \subsetneq y$, then there is there is a finite sequence of atomic extension*

$$x \xrightarrow{e_1} \subset x_1 \xrightarrow{e_2} \subset \cdots \xrightarrow{e_n} \subset y.$$

Proof. Let e be a minimal event in $y \setminus x$ for the partial order \leq_y , then, by definition, $x \xrightarrow{e} \subset$ defines an atomic extension. One can thus repeat this process $|y \setminus x|$ times to get the desired chain. \square

On figure 1.3 we present an event structure (left) together with its partial order set of configurations (right), organised following its covering relation. This characterisation of configurations as chains of atomic extension gives a process flavor to event structures: configurations are the observational result of some sequences of events that respect causality and consistency.

Constructions on event structure Similarly to processes, event structures enjoy three majors constructions: synchronous product, asynchronous product (or parallel composition) and projection [Win86]. We focus on the last two, saving the first one for later.

For U, V two sets, we denote $U \uplus V$ their (tagged) *disjoint union* $\{0\} \times U \cup \{1\} \times V$. We set:

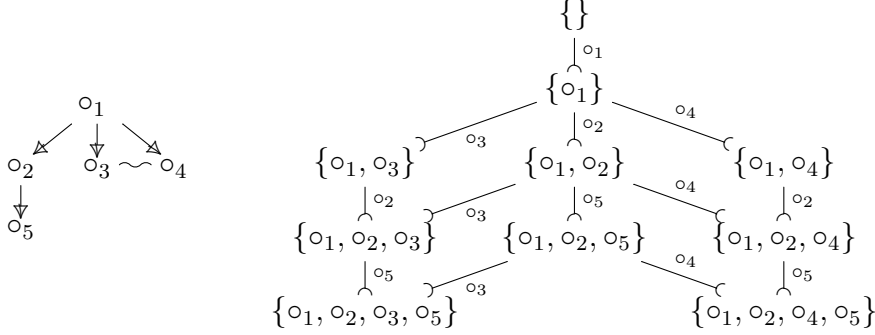


Figure 1.3: An event structure and its (partially ordered) set of configurations

Definition 1.2. Let E_1, E_2 be event structures, then their *parallel* composition $E_1 \parallel E_2$ is the event structure defined by:

- Events: $E_1 \parallel E_2 = E_1 \uplus E_2$,
- Consistency: $\text{Con}_{E_1 \parallel E_2} = \{X_1 \uplus X_2 \mid X_1 \in \text{Con}_{E_1}, X_2 \in \text{Con}_{E_2}\}$,
- Enabling: $(i, e) \leq_{E_1 \parallel E_2} (j, e')$ iff $i = j$ and $e \leq_{E_i} e'$.

Every configuration x in $\mathcal{C}(E_1 \parallel E_2)$ is of the form $(x_1 \parallel x_2)$ where $x_1 \in \mathcal{C}(E_1)$ and $x_2 \in \mathcal{C}(E_2)$, denoted $x_1 \parallel x_2$. Hence, the set-inclusion on configurations matches the set-inclusion of their component.

Definition 1.3. Let E be an event structure and $V \subseteq E$, then the *projection of E onto V* is the event structure $E \downarrow V$ defined by:

- Events: $E \downarrow V = V$,
- Consistency: $\text{Con}_{E \downarrow V} = \{X \cap V \mid X \in \text{Con}_E\}$,
- Enabling: $v \leq_{E \downarrow V} v'$ iff $v \leq_E v'$.

In the definition above, V is often referred as the *visible* set of events. Indeed, projection can be understood as hiding away events that are no longer observable, namely the ones in the complement of V . Configurations of $E \downarrow V$ can also be described using configuration of E : if $x \in \mathcal{C}(E)$ then $x \cap V \in \mathcal{C}(E \downarrow V)$. However, this description is not unique: a configuration $x \in \mathcal{C}(E \downarrow V)$ can have several *witnesses* $x' \in \mathcal{C}(E)$ such that $x = x' \cap V$. To get back a correspondence, one restricts to *visible configurations*, that are configurations of E whose maximal events (with respect to \leq_E) are visible. We have

Lemma 1.3. *Let E be an event structure and $V \subseteq E$, then for every configuration $x \in \mathcal{C}(E \downarrow V)$ there is a unique a visible configuration, called the minimal witness of x , denoted $\text{wit}_E(x) \in \mathcal{C}(E)$, such that $\text{wit}_E(x) \downarrow V = x$.*

Moreover, set-inclusion over minimal witnesses matches the one in $\mathcal{C}(E \downarrow V)$.

Morphisms Event structures are equipped with a notion of morphisms that intuitively correspond to simulations of processes by preserving configurations and event occurrences:

Definition 1.4. Let E, F be two event structures, a function $f : E \rightarrow F$ is a *morphism of event structures* if it

(i) preserves configurations: $\forall x \in \mathcal{C}(E), f(x) \in \mathcal{C}(F)$;

(ii) is *locally injective*: $\forall e, e' \in x \in \mathcal{C}(E)$, if $f(e) = f(e')$ then $e = e'$.

In the above, f describes a renaming of the events of E to events in F so that (condition (i)) E can be viewed as a causal and conflict enrichment of F . In that regard, condition (ii) ensures that every event that occurs in E will necessarily be witnessed in F : for every configuration $x \in \mathcal{C}(E)$, the restriction of f to x is a bijection

$$f|_x : x \simeq f(x)$$

In other words, two distinct events in E , can only be mapped to the same event in F if they are inconsistent with each other.

As an example, the event structure E from figure 1.1 can be mapped to the event structure F on figure 1.2 in nine different ways: sending \circ_1 to either **dough**, **tomatoes**, **pasta**, then sending events $\circ_2, \circ_3, \circ_4$ to any permutation of **salad**, **spaghetti**, **pizza**.

Other simple examples of event structure morphisms are the identity functions, the functional composition of event structure morphisms as well as the parallel composition of morphisms (defined component-wise as expected). In fact, it is routine to check that:

Theorem 1.1. *Event structures and their morphisms define a category called \mathcal{E} . It has a monoidal structure given by (\parallel, \emptyset) , where \emptyset is the event structure with no events.*

We conclude this section by emphasizing the connection between event structures and their configurations with a lemma that relates the two:

Lemma 1.4 (Lemma 2.17 in [CCRW17]). *Let A and B be event structures, then there is a one to one correspondence between order-isomorphisms $\varphi : \mathcal{C}(A) \cong \mathcal{C}(B)$ and isomorphisms $\hat{\varphi} : A \cong B$, such that*

$$\hat{\varphi}(x) = \varphi(x)$$

for every configuration $x \in \mathcal{C}(A)$.

Proof sketch. From φ , for $a \in A$, $\hat{\varphi}(a)$ is defined as the unique event in $\varphi([a]) - \varphi([a])$. From $\hat{\varphi}$ to φ one simply takes $\varphi(x) = \varphi(\hat{\varphi}(x))$. One can then show that two parallel morphisms of event structures are equal if and only if they are equal on their set of configurations, concluding the proof of uniqueness. \square

1.2 Category of concurrent games and strategies

In this section we recall how the category of event structures described above can be used as a basis to define CG, the category of concurrent games and strategies.

1.2.1 Games and strategies as event structures with polarities

As briefly mentioned in section 1.1, event structures were first introduced in semantics to give an unfolded representation of non-deterministic and concurrent transition systems such as Petri nets [Win86]. There, inconsistency describes non-determinism and configurations match states of the system.

Following this idea, the concurrent games model based on event structures takes games as event structures and set configurations as game positions. But game semantics is a semantics of interaction and open terms; it has a notion of players and strategies for those players.

Definition 1.5. An *event structure with polarities*, also called a *game*, is an event structure A along with a function

$$\text{pol}_A : A \rightarrow \{-, +\}$$

The polarity function assigns each event, also called *move*, to a given player. We write $A^+ = \{a \in A \mid \text{pol}_A = +\}$ for the set of Player moves, and $A^- = \{a \in A \mid \text{pol}_A = -\}$ for the set of Opponent moves.

$$\begin{array}{c}
\llbracket \mathbf{mem} \rrbracket = \mathbf{mem}_W \otimes \mathbf{mem}_R \\
\begin{array}{ccc}
\mathbf{wtt}^- & & \mathbf{r}^- \\
\vdots & & \vdots \\
\mathbf{ok}^+ & \mathbf{tt}^+ \text{---} \mathbf{ff}^+ &
\end{array}
\end{array}$$

Figure 1.4: The composed game of memory cell.

Definition 1.6. Every game A has a *dual* A^\perp that corresponds to A with reversed polarities.

The monoidal category of event structures extends straightforwardly to polarities. Morphisms are those that preserve polarities and the extra polarity component is treated as expected in parallel composition and projection. We note \mathcal{EP} for the corresponding category.

An example game is shown on figure 1.4. It depicts the game representing the type of a memory cell. In this game Opponent has three moves available: \mathbf{r} , \mathbf{wtt} , \mathbf{wff} to respectively perform a read, a write true or a write false request to the memory. Each of these moves enables in turn one or two appropriate answers for Player. Contrary to sequential games, players do not necessarily alternate when playing on a concurrent game. In the previous example, configuration of $\llbracket \mathbf{mem} \rrbracket$ may allow Opponent to play its three moves in a row before Player actually gets to answer. Yet it may also happen that Opponent and Player play one after the other. With no further restriction, consistency and the causality relation can thus be seen as the rules of the game.

Strategies. Generally in game semantics, strategies describe how players choose their next moves from a given position (if any available). This amounts to selecting a way (or several ways in case of concurrency/non-determinism) to navigate from one position to an other, following the rules of the game. In concurrent games, strategies are implemented using morphisms of event structures with polarities together with some constraints.

Definition 1.7. A *strategy for Player* on a game A is a morphism of event structures with polarities $\sigma : S \rightarrow A$ that satisfies two sanity conditions:

- (i) *receptivity*: $\forall x \in \mathcal{C}(S)$, if $\sigma(x) \xrightarrow{a^-}$ then there exists a *unique* $s \in S$ (necessarily negative) such that $x \xrightarrow{s}$ and $\sigma s = a$;
- (ii) *courtesy*: $\forall s, s' \in S$ such that $s \rightarrow_S s'$ and $(\text{pol}(s), \text{pol}(s')) \neq (-, +)$, then $\sigma s \rightarrow_A \sigma s'$.

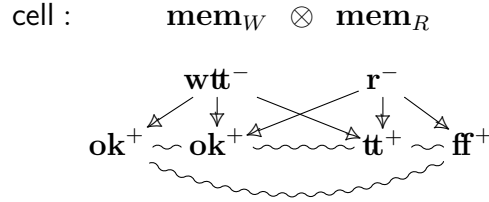


Figure 1.5: A strategy for memory cell

The event structure S is called the *support* of σ .

Figure 1.5 depicts a strategy that plays on the memory game from figure 1.4; rather than giving the explicit mapping from its support to $\llbracket \mathbf{mem} \rrbracket$, we directly label the events of \mathbf{cell} with names from the move of the game. More generally, a strategy $\sigma : S \rightarrow A$ can be viewed as a labelling of the event in its support S with the moves in its targeted game A .

By relabelling the configurations of S , σ “selects positions” in A , then, the fact that σ is a morphism ensures that the chosen positions really define positions/configurations in A (def 1.4 (i)), hence that σ obeys the rules of A .

That σ is a morphism also ensures that it plays linearly with respect to A : by condition (ii) in definition 1.4, two moves in S are labelled with the same move in A only if they are inconsistent with each other. Such duplications are often used in a strategy to *internally* remember some contextual information or branching point. For example, on figure 1.5 the strategy mimics the behaviour of a memory cell that has two ways of acknowledging a write (with the \mathbf{ok} move) depending on whether a read had already been performed or not.

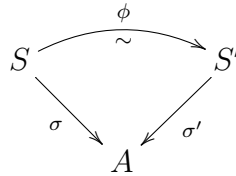
Receptivity and Courtesy in definition 1.7 put even more constraints on the shape of S . From a game semantics point of view these constraints remain intuitive:

- (i) Receptivity ensures that a strategy for Player only has choices upon Player moves: it must accept every move Opponent can play according to the rules of the game. Uniqueness in receptivity stands for sanity: it prevents Player from internally distinguishing an Opponent move according for example to the context in which it occurs.
- (ii) Courtesy requires that the only new immediate causal dependencies introduced in S with respect to A are from negative moves to positive moves. This follows the intuition that a player reacts to its opponent’s moves. This also has to do with the *asynchronous* nature of concurrent games: rather than playing face to face, one can imagine that in these models the two players play several miles away. Positive and negative

events can then be respectively thought as sending and receiving notifications and the order in which these notifications arrive or are received is not guaranteed except if it is specified in the game itself. Hence, in order to stay synchronised with its opponent, a player can only delay its own (positive) moves relatively to its opponent's (negative) move.

It should be clear from the above that the name of the events in the support of a strategy does not really matter; what matters is their labelling as moves in the targeted game, as well as their consistency and causality relation. To strengthen this point, strategies are usually treated up to isomorphisms:

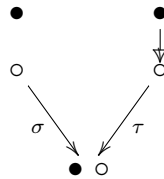
Definition 1.8. Two strategies $\sigma : S \rightarrow A$ and $\sigma' : S' \rightarrow A$ are *isomorphic*, written $\sigma \simeq \sigma'$, if there exists an isomorphism $\phi : S \cong S'$ such that the following diagram commutes:



1.2.2 Interaction of strategies.

Definition 1.7 introduces what are strategies for Player on a game A . *Strategies for Opponent* (also called *counter-strategies*) are defined dually: they are strategies for Player on the dual game A^\perp . Strategies and counter strategies can *interact*. Given two strategies $\sigma : S \rightarrow A$ and $\tau : T \rightarrow A^\perp$, their interaction, denoted $S \wedge T$ is closely related to the synchronous product for processes mentioned in the previous section; it relies on the fact that the underlying category \mathcal{E} has *pullbacks*.

Intuitively, two strategies $\sigma : S \rightarrow A$ and $\tau : T \rightarrow A^\perp$ interact with each other by pairwise *synchronising* their moves. Two moves can be synchronised if they are mapped to the same event in the game A and if they are available in both of the strategies by the time they are played. For example, in the picture below, the two \circ events can synchronise but they can only do so after the two \bullet events did, as otherwise \circ is not available in τ .



Configurations of the interaction, are thus obtained from the empty configuration by successively adding *synchronised pairs of events*, that are, pairs $(s, t) \in S \times T$ such that $\sigma(s) = \tau(t)$, with the sanity condition that s and t are both available in their respective strategy.

A first attempt to characterise the configurations in $S \wedge T$ is to take the *synchronised pairs of configurations*, that are pairs of configurations $(x_S, x_T) \in \mathcal{C}(S) \times \mathcal{C}(T)$ such that $\sigma(x_S) = \tau(x_T)$. However, adding an immediate causality $\circ \rightarrow \bullet$ in the strategy σ in the above example, the pair $(\{\bullet, \circ\}, \{\bullet, \circ\})$ would still define a valid synchronised pair in that sense, but impossible to obtain from the empty configuration by adding one move at a time. To exclude such a pattern, one further requires that the bijection $x_S \xrightarrow{\sigma} \sigma(x_S) = \tau(x_T) \xrightarrow{\tau^{-1}} x_T$ induced by a synchronised pair of configurations (x_S, x_T) is *secured*:

Definition 1.9. The bijection $\varphi : x_S \simeq x_T$ induced by a pair of synchronised configurations $(x_S, x_T) \in \mathcal{C}(S) \times \mathcal{C}(T)$ is *secured* if the reflexive and transitive closure of

$$(s, t) \leq (s', t') \quad \text{iff} \quad s \leq_S s' \vee t \leq_T t'$$

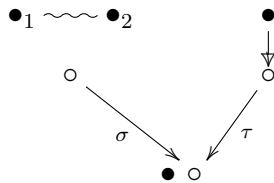
defines an order on the graph of φ , denoted \leq_φ .

That φ is secured means that there are no loops in the graph of causality induced by \leq_{x_S} and \leq_{x_T} , so \leq_φ can be linearised into a sequence of synchronised moves.

The above gives a description of the set of configurations expected in the interaction of σ and τ . From that it is possible to recover the corresponding event structure $S \wedge T$. Let us sketch this construction.

First, one may be tempted to take synchronised pair of events as events for $S \wedge T$. However, because of possible conflicts in S and T , the same synchronised pair can have several (incompatible) causal histories in the interaction.

Example 1.1. Let $A = \{\bullet, \circ\}$ and let σ and τ be defined by:



Then the configurations of $S \wedge T$ are expected to correspond to $\emptyset, \{(\bullet_1, \bullet)\}, \{(\bullet_2, \bullet)\}, \varphi_1 = \{(\bullet_1, \bullet), (\circ, \circ)\}, \varphi_2 = \{(\circ_1, \bullet), (\circ, \circ)\}$. The synchronised pair (\circ, \circ) appears twice with two different causal histories.

In the example above, the two versions of (\circ, \circ) must be distinguished. A simple way for that is to annotate every synchronised pair of events by the secured bijection corresponding to its causal history. In the above we would get $[\circ, \circ]_{\varphi_1}$ and $[\circ, \circ]_{\varphi_2}$.

To be more precise, let

$$\mathcal{B}_{\sigma, \tau}^{\text{sec}} = \{\varphi : x_S \simeq x_T \mid \varphi \text{ a secured bijection for } \sigma, \tau\}$$

be the set of secured bijections induced by σ and τ , then a secured bijection $\varphi \in \mathcal{B}_{\sigma, \tau}^{\text{sec}}$ is said to be *prime* if it contains a unique maximal synchronised pair of event for \leq_{φ} , denoted $[\![s, t]\!]_{\varphi}$. Although we drew intuition from the interaction of strategies, interaction is defined for every two morphisms of event structures sharing the same codomain:

Definition 1.10. Let $\sigma : S \rightarrow A$, $\tau : T \rightarrow A$ be two maps of event structures, their interaction $S \wedge T$ is the event structure defined by:

- Events: the prime secured bijections of $\mathcal{B}_{\sigma, \tau}^{\text{sec}}$,
- Enabling: inclusion of the graphs of the secured bijections,
- Consistency: X is a consistent set of prime secured bijections if the union of their graphs describes a secured bijection (*i.e.* $(\bigcup_{\varphi \in X} \varphi) \in \mathcal{B}_{\sigma, \tau}^{\text{sec}}$).

The interaction $S \wedge T$ comes with two obvious projections $\Pi_1 : S \wedge T \rightarrow S$ and $\Pi_2 : S \wedge T \rightarrow T$, that are morphisms of event structures. Moreover, by definition, these maps make the following diagram commutes in \mathcal{E} :

$$\begin{array}{ccc} & S \wedge T & \\ \Pi_1 \swarrow & & \searrow \Pi_2 \\ S & & T \\ \sigma \searrow & & \swarrow \tau \\ & A & \end{array}$$

In fact we have the following proposition:

Proposition 1.1. *Let $\sigma : S \rightarrow A$ and $\tau : T \rightarrow A$ be two morphisms of event structures, then the construction $(S \wedge T, \Pi_1, \Pi_2)$ defines a pullback in \mathcal{E} .*

We write $\sigma \wedge \tau : S \wedge T \rightarrow A$ for the morphism of event structures given by $\sigma \circ \Pi_1 = \tau \circ \Pi_2$.

We do not detail the proof here as we will reprove this statement in proposition 3.2 for the more general case of morphisms of annotated event structures. We conclude however on an intermediate lemma that makes explicit the strong correspondence between configurations in $\mathcal{C}(S \wedge T)$ and secured bijections. This is a restatement of Lemma 2.9 in [CCRW17].

Lemma 1.5. *For any configuration $x \in \mathcal{C}(S \wedge T)$, x defines a secured bijection $\varphi_x = \cup x : \Pi_1 x \simeq \Pi_2 x$. Moreover, the assignment $x \mapsto \varphi_x$ defines an order-isomorphism $\mathcal{C}(S \wedge T) \cong \mathcal{B}_{\sigma, \tau}^{\text{sec}}$ (with both sets ordered by inclusion), and there is a family of order-isomorphisms:*

$$\begin{array}{ccc} \nu_x & : & x \quad \simeq \quad \varphi_x \\ & & [s, t]_x \mapsto (s, t) \end{array}$$

that is natural in x .

For $\varphi \in \mathcal{B}_{\sigma, \tau}^{\text{sec}}$ we also write $x_\varphi \in \mathcal{C}(S \wedge T)$ for the configuration such that $\varphi_{x_\varphi} = \varphi$.

Proof. By definition of consistency in $S \wedge T$, $\cup x$ is the graph of a secured bijection. This mapping obviously preserves inclusion and its converse maps a secured bijection φ to the set of elements of $S \wedge T$ included in φ . Then, by definition of causality, ν_x also defines an order-isomorphism and naturality is by definition. \square

This lemma is very useful for reasoning on configurations of an interaction as it allows us to manipulate secured bijections rather than compatible sets of prime secured bijections. This correspondence is precise enough that reasoning on the events of the interaction in an ambient configuration, is the same as reasoning on the synchronised pairs of the corresponding secured bijection, transporting the reasoning through ν . In the sequel, we will often make use of this correspondence and transfer silently between the two representations.

1.2.3 Composition of strategies

The previous section presents how strategies and counter-strategies over a game A can interact. This construction is a bit rigid as it requires the two strategies to play over the exact same game (with reverse polarities). What if two strategies play over compound game and only share some dual components? This question leads to view strategies as playing from one game to an other.

Definition 1.11. A strategy is said to play *from a game A to a game B* if it is of the form

$$\sigma : S \rightarrow A^\perp \parallel B$$

When its support is not relevant, σ is simply noted $\sigma : A \dashrightarrow B$.

This definition implements the fact that σ is playing as Opponent on A and as Player on B .

Copycat strategy. Given a game A , a standard example of strategy from one game to another is the copycat strategy $\mathfrak{c}_A : A \multimap A$. As in sequential games, this strategy propagates Opponent moves from one component to its corresponding Player move in the other component. Formally

Definition 1.12. Let A be a game, then $\mathfrak{c}_A : A \multimap A$ is the strategy

$$\mathfrak{c}_A : \mathbb{C}_A \xrightarrow{\text{id}_{A^\perp \parallel A}} A^\perp \parallel A$$

where \mathbb{C}_A is the event structure described by:

- Events: those of $A^\perp \parallel A$,
- Enabling: the reflexive and transitive closure of

$$\leq_{A^\perp \parallel A} \cup \left\{ (i, a) \rightarrow (1-i, a) \mid (i, a) \in (A^\perp \parallel A)^- \right\}$$

- Consistency: $X \in \text{Con}_{\mathbb{C}_A}$ iff $[X]_{\mathbb{C}_A} \in \text{Con}_{A^\perp \parallel A}$.

An example of copycat strategy is depicted in the middle of figure 1.6 for the game $A = \bullet^- \circ^+$.

Composition. Given two strategies $\sigma : A \multimap B$ and $\tau : B \multimap C$, their *interaction* is the morphism of event structures

$$\tau \otimes \sigma = (\sigma \parallel C) \wedge (A \parallel \tau) : T \otimes S \rightarrow A \parallel B \parallel C$$

where C and A stand for the identity morphisms on C and A , and $T \otimes S$ is a shortening for $(S \parallel C) \wedge (A \parallel T)$.

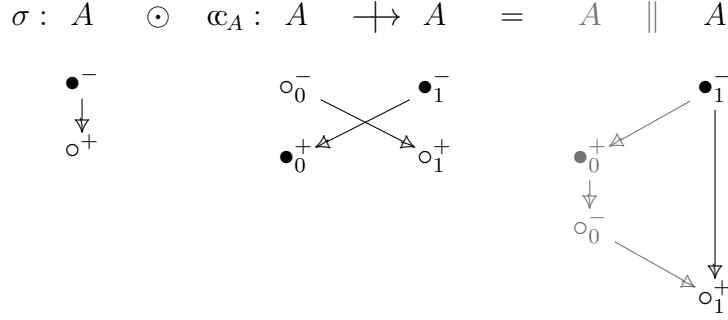
From the interaction $\tau \otimes \sigma : T \otimes S \rightarrow A \parallel B \parallel C$, one can get back a strategy by hiding away events that correspond to moves in the common component B and putting back polarities.

Definition 1.13. Let $\sigma : A \multimap B, \tau : B \multimap C$ be two strategies, the *composition* of σ and τ is the strategy $\tau \odot \sigma$ defined by

$$\tau \odot \sigma = (\tau \otimes \sigma) \downarrow V : T \odot S \rightarrow A^\perp \parallel C$$

where $V \subseteq T \otimes S$ is the set of events in $T \otimes S$ that map to the outer games $A \parallel C$ and $T \odot S$ is a shortening for $(T \otimes S) \downarrow V$ with polarities inherited from those of $A^\perp \parallel C$.

Example 1.2. Figure 1.6 shows the composition of a strategy $\sigma : A$ with the copycat strategy $\mathfrak{c}_A : A \multimap A$ described earlier. Their interaction is left visible in grey but the result after hiding lies under the rightmost game A . One can note that (up to isomorphism), this is exactly σ .


 Figure 1.6: Composition with the copycat strategy for $A = \bullet^- \circ^+$

In what follows, we will see that in fact \mathfrak{c}_A is neutral for composition on A . We will also review why composition of strategies is associative; turning games and strategies (up to isomorphism) into a category.

Before moving to these two points we introduce a couple more useful notations. Following lemma 1.5, the configurations of $T \otimes S$ and their causal order are described by secured bijections $\varphi : x_S \parallel x_C \simeq x_A \parallel x_T$ such that $\sigma(x_S) = x_A \parallel x_B$ and $\tau(x_T) = x_B \parallel x_C$. Eliminating redundancy, φ can be subsumed into the data of x_S and x_T . By extension, we will write $x_T \otimes x_S$ for the corresponding configuration in $\mathcal{C}(T \otimes S)$.

Similarly, following lemma 1.3, configurations in $T \odot S$ correspond to visible configurations in $T \otimes S$. By lemma 1.5 again, these correspond to secure bijections that are *visible*, *i.e.* such that their maximal synchronised pair of events (with respect to \leq_φ) are mapped to $A \parallel C$ (though $\sigma \circ \pi_1 = \tau \circ \pi_2$). These bijection can still be characterised by the data of their corresponding x_S and x_T . In that case, we write $x_S \odot x_T$ for the corresponding configuration in $\mathcal{C}(T \odot S)$. Note that in both case, the partial order on configurations is reflected in the partial order of the components.

With these notations, it is immediate to see that the isomorphic relation \simeq on strategies is a congruence with respect to the composition:

Proposition 1.2. *Let $\sigma, \sigma' : A \dashv \rightarrow B$ and $\tau, \tau' : B \dashv \rightarrow C$ such that $\sigma \simeq \sigma'$ and $\tau \simeq \tau'$, then*

$$\sigma \odot \tau \simeq \sigma' \odot \tau'$$

Proof. Let $\varphi_1 : S \cong S'$, $\varphi_2 : T \cong T'$ be the isomorphisms of event structures given by $\sigma \simeq \sigma'$ and $\tau \simeq \tau'$. Then, following the remarks above there is an order isomorphism

$$\begin{aligned} \varphi : \mathcal{C}(T \odot S) &\cong \mathcal{C}(T' \odot S') \\ x_T \odot x_S &\mapsto \varphi_1(x_T) \odot \varphi_1(x_S) \end{aligned}$$

Moreover $\hat{\varphi}$ commutes with $\sigma \odot \tau$ and $\sigma' \odot \tau'$, concluding the proof. \square

Associativity. Following [CCRW17], proving associativity for composition requires technicalities that are unnecessary for the presentation of concurrent games we are doing. We only sketch:

Proposition 1.3. *Let $\sigma : A \rightarrow B$, $\tau : B \rightarrow C$, $\rho : C \rightarrow D$ then*

$$\rho \odot (\tau \odot \sigma) \simeq (\rho \odot \tau) \odot \sigma$$

Proof sketch. This relies on the fact that interactions are pullbacks in \mathcal{E} . By universal property, this induces a morphism of event structure such that the following diagram commutes

One can then show that hiding on B preserves this diagram and conclude. \square

Composition with copycat. Although we did not emphasize it, let us note that the composition of strategies and its proof of associativity actually holds for any morphisms of event structures with polarities of the form $\sigma : S \rightarrow A^\perp \parallel B$, without the receptivity and courtesy conditions to be necessary. In \mathcal{EP} these morphisms are sometimes called *pre-strategies from A to B* precisely because they support associative composition.

Courtesy and receptivity however are necessary to distinguish among pre-strategies those for which the copycat strategies from definition 1.12 are neutrals for composition. To get a first intuition for this fact, let us study the shape of the configurations of \mathbb{C}_A . We first give a more explicit characterisation of its causal structure:

Lemma 1.6. *For $(i, a), (j, a') \in \mathbb{C}_A$, $(i, a) \leq_{\mathbb{C}_A} (j, a')$ iff*

- $i = j$ and $a \leq_A a'$, or
- $i = 0, j = 1$ and $a \in [a']_{A^*}$; or
- $i = 1, j = 0$ and $a \in [a']_{A^*}$.

where $[a]_A^*$ is the minimal configuration to reach $[a]_A$ with only negative extensions (i.e. $[a]_A^* = \min\{x \subseteq^- [a] \mid x \in \mathcal{C}(A)\}$) and similarly for $[a]_{A,*}$ and positive extensions.

Proof. The first case is trivial by definition of \mathbb{C}_A . The two other cases have similar proofs so we focus on the second one. The left implication is direct following the definition of \mathbb{C}_A : $a' \in [a]_A^*$ implies that there exists $a'' \in A^+$ such that $a' \leq_A a'' \leq_A a$ and so $(0, a) \leq_A (0, a'') \rightarrow (1, a'') \leq_A (1, a')$ in \mathbb{C}_A .

For the right implication, first recall that $(0, a) \leq_{\mathbb{C}_A} (1, a)$ is equivalent (cf.p. 21) to having a chain of immediate causal dependencies

$$(0, a) \rightarrow (i_1, a_1) \rightarrow \cdots \rightarrow (1, a')$$

where $\rightarrow = \rightarrow_{A^+ \parallel A}$ or $(1 - i_j, a_j)^- \rightarrow (i_{j+1}, a_{j+1})^+$. Forgetting the information for left and right component this gives $a \rightarrow a_1 \rightarrow \cdots \rightarrow a'$ where $\rightarrow = \rightarrow_A$ or $a_{j+1} = a_j$, hence $a \in [a']_A$. Then crossing from left to right component implies that there is at least one immediate causal dependency of the form $(0, a_j) \rightarrow (1, a_j)$ with $\text{pol}(a_j) = +$, hence $a \leq_A a_j^+ \leq_A a'$ and so $a \in [a']^*$. \square

Proposition 1.4. *Configurations in \mathbb{C}_A are characterized by*

$$x_0 \parallel x_1 \in \mathcal{C}(\mathbb{C}_A) \quad \text{iff} \quad x_0, x_1 \in \mathcal{C}(A) \quad \text{and} \quad x_1^- \supseteq x_0 \cap x_1 \subseteq^+ x_0$$

where \subseteq^+ corresponds to extension with only positive moves, and similarly for \subseteq^- and negative moves.

The relation $^- \supseteq \subseteq^+$ defines a partial order, denoted \sqsubseteq for short.

Proof. (\Rightarrow) That $x_0, x_1 \in \mathcal{C}(A)$ is immediate by consistency and causality in \mathbb{C}_A . Then the other condition followed by lemma 1.6 and down-closure.

(\Leftarrow) Two configurations $x_0, x_1 \in \mathcal{C}(A)$ are necessarily consistent in \mathbb{C}_A . By lemma 1.6 the condition $x_1 \sqsubseteq x_0$ also ensures that $x_0 \parallel x_1$ is down-closed. \square

The proposition above shows that the configurations in the left and right components of copycat are almost mirrors of each other but not quite. There might be a mismatch on Opponent moves. In fact, in concurrent games, copycat strategies are sometimes referred as *asynchronous forwarder* as they introduce uncontrolled delays while recasting Opponent moves: there is no guarantee that the order in which Opponent moves are received is preserved by the recast. That is the reason why, on top of being receptive, strategies must be courteous – i.e. they must not assume anything else on the order of negative events than what is imposed by the rules of the game.

In the rest of this section we recast the proof that copycat indeed acts as the identity for the composition of strategies; the argument will be needed

to extend the model with annotations. However we refrain to explain further the characterisation of strategies as receptive and courteous pre-strategies is exact, referring to [CCRW17] for additional details.

Proposition 1.5. *Let $\sigma : A \rightarrow B$ be a strategy, then $\sigma \odot \mathbb{C}_A \simeq \sigma \simeq \mathbb{C}_B \odot \sigma$.*

Proof. We focus on the case where $\sigma : S \rightarrow A$ is post-composed by \mathbb{C}_A , the proof for pre-composition is similar and the proof for the more general case is just more technical.

Following definition 1.8, the goal is to exhibit an isomorphism $\chi : S \cong \mathbb{C}_A \odot S$ that commutes with σ and $\mathbb{C}_A \odot \sigma$. With this in mind, let us show that the mapping

$$\begin{aligned} \chi : S &\xrightarrow{\sim} \mathbb{C}_A \odot S \\ s &\mapsto \varphi_{(\sigma([s]^*) \parallel \sigma[s]) \odot [s]^*} \end{aligned}$$

does exactly the trick – recall that φ_x is the secured bijection associated to a configuration $x \in \mathbb{C}_A \otimes S$, hence defines an event in $\mathbb{C}_A \odot S$ if the corresponding configuration is prime.

The hardest part is to show that the above mapping actually is well-defined and bijective. If it does, the rest follows: by definition $\sigma = (\mathbb{C}_A \odot \sigma) \circ \chi$ and the corresponding isomorphism on configuration clearly preserves \subseteq hence, by lemma 1.4, χ defines an isomorphism of event structures with polarities.

Let us first check that χ is well-defined, that is, $\sigma([s]^*) \parallel \sigma[s]$ and $[s]^*$ are valid data to define a prime secured bijection. Clearly $[s]^* \in \mathcal{C}(S)$ and $\sigma([s]) \sqsubseteq_A \sigma([s]^*)$ so, by proposition 1.4, $\sigma([s]^*) \parallel \sigma[s] \in \mathcal{C}(\mathbb{C}_A)$. Moreover, the bijection

$$\varphi : [s]^* \parallel \sigma[s] \simeq (\sigma([s]^*) \parallel \sigma[s])$$

is secured as the partial ordering of \mathbb{C}_A on A_0 is the one of A , hence is necessarily aligned with the one of S . It remains to prove that it is prime. Writing $(s, \sigma(s))$ and (a, a) respectively for the synchronised pairs of events $((0, s), (0, \sigma(s)))$ and $((1, a), (1, a))$ in φ , we show that φ has top event $e = (\sigma(s), \sigma(s))$, as every other event $e' \neq e$ in φ is dominated, by case analysis:

- for $e' = (s', \sigma(s'))$ then $s' \in [s]^*$, if s' is not maximal in $[s]^*$ then e' is not maximal in φ via \leq_S , if it is maximal, then s is positive and so $e' \leq (\sigma(s'), \sigma(s'))$ via $\leq_{\mathbb{C}_A}$;
- for $e' = (\sigma(s'), \sigma(s'))$, then $s' \in [s]$. Now if s' negative and $s' \in [s]^*$ then $e' \leq (s', \sigma(s'))$, via $\leq_{\mathbb{C}_A}$ otherwise $s' \rightarrow_S s'' \leq s$ with s'' negative so, by courtesy, $\sigma(s') \rightarrow_A \sigma(s'')$ and so $e' \leq (\sigma(s''), \sigma(s''))$ via $\leq_{\mathbb{C}_A}$, finally if s' is positive then $s' \rightarrow_S s'' \in [s]$ so by courtesy $e' \leq (\sigma(s''), \sigma(s''))$ via $\leq_{\mathbb{C}_A}$.

Finally we show that every prime secured bijection $\varphi : x_S \parallel x_A \simeq \sigma(x_S) \parallel x_A$ that is visible, *i.e.* whose maximal event is of the form (a, a) , is actually of the form $(\sigma([s]^*) \parallel \sigma[s]) \odot [s]^*$. First note that by proposition 1.4, $x_A \sqsubseteq \sigma(x_S)$ so, by receptivity, there exists a unique $x'_S \in \mathcal{C}(S)$, such that $x_S \sqsubseteq^- x'_S$ and $x_A \sqsubseteq^+ \sigma(x_S)$. Moreover, a simple induction on φ shows that the mapping

$$f : \begin{array}{lll} \varphi & \rightarrow & x'_S \\ (s, \sigma(s)) & \mapsto & s \\ (a, a) & \mapsto & \sigma_{x'_S}^{-1}(a) \end{array}$$

preserves the partial order on φ . In particular events that correspond to negative extensions in x_A and positive extensions in x_S are incomparable with each other in φ (using courtesy in the second case)

Now, if φ is prime with top event $e = (a, a)$ then this remark implies that $\sigma(x_S) \sqsubseteq^- x_A$ and that x'_S is prime, so $x'_S = [s]$ for some $s \in S$. Moreover $x_S = (x'_S)^*$ – as otherwise this would contradict the maximality of e in x . Summing up, this leads to $(x_S, x_A) = ([s]^*, \sigma([s]))$. \square

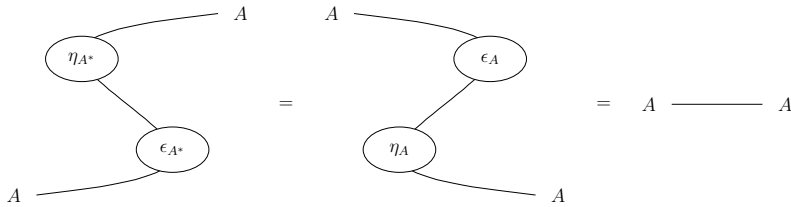
1.3 Compact closed structure

The previous section introduces concurrent games and strategies as described in [CCRW17] and shows that – up to isomorphism – they form a category called CG. We now recall the result establishing that CG actually is a compact closed category.

As a reminder, a compact closed category is a symmetric monoidal category $(\mathcal{C}, \otimes, \mathbf{1})$ in which every object A has a *dual* A^* for which there exist two morphisms (called the unit and co-unit)

$$\eta_A : \mathbf{1} \rightarrow A^* \otimes A \qquad \epsilon_A : A \otimes A^* \rightarrow \mathbf{1}$$

such that the following equalities hold



From a semantic point of view, compact closed models are desirable as they come with a canonical interpretation of MLL on the logical side, and of the linear lambda-calculus on the programming language side. In the sequel we

will ensure that this categorical structure is always preserved by our various extensions.

In CG, the tensor product is lifted from the monoidal structure of \mathcal{EP} and defined on games by $A \otimes B = A \parallel B$ with neutral element the empty game $\mathbf{1} = \emptyset$.

For duality, A^* is taken to be the dual game A^\perp and η_A and ε_A correspond to “curried” versions of copycat

$$\begin{aligned}\eta_A &: \mathbb{C}_A \rightarrow \mathbf{1} \parallel (A^\perp \parallel A) \\ \varepsilon_A &: \mathbb{C}_A \rightarrow (A \parallel A^\perp)^\perp \parallel \mathbf{1}\end{aligned}$$

This is not surprising as dual objects of a compact closed category can be viewed as defining an adjunction

$$\frac{B \multimap A^* \otimes C}{A \otimes B \rightarrow C}$$

and the unit and co-unit mentioned above are respectively the image of id_A and id_{A^*} through this adjunction.

We refrain from recasting the proof of [CCRW17] showing that the η_A and ε_A strategies satisfy the law of compact closure. Yet chapter 3 will be dedicated to show that the extension of CG with annotations also has a compact closed structure and will follow the same proof techniques as in [CCRW17] hence introducing the technical details omitted here. We conclude this section by detailing the symmetric monoidal structure of CG, sketching some proofs from [CCRW17] and leaving others to chapter 3.

Symmetric monoidal structure CG is equipped with a bifunctor \otimes that performs parallel composition on games and strategies. More precisely $A \otimes B = A \parallel B$ and for $\sigma : S \rightarrow A^\perp \parallel C$, $\tau : T \rightarrow B^\perp \parallel D$

$$\sigma \otimes \tau : S \parallel T \xrightarrow{\sigma \parallel \tau} A^\perp \parallel C \parallel B^\perp \parallel D \xrightarrow{\gamma} A^\perp \parallel B^\perp \parallel C \parallel D$$

where $\gamma : A^\perp \parallel C \parallel B^\perp \parallel D \rightarrow A^\perp \parallel B^\perp \parallel C \parallel D$ is the obvious isomorphism. We write $x_S \otimes x_T$ for the configuration corresponding to $\gamma(x_S \parallel x_T)$ in $\mathcal{C}(S \otimes T)$. Every configuration is of this form and \subseteq on $\mathcal{C}(S \otimes T)$ follows \subseteq on the two components.

The following two lemmas justify that \otimes is well-defined on strategy and that it preserves the isomorphic relation.

Lemma 1.7. *Let $\gamma : A \cong B$ be an isomorphism of games and $\sigma : S \rightarrow A$ be a strategy, then $\gamma \circ \sigma : S \rightarrow B$ also defines a strategy.*

Proof sketch. One can first check that the composition of courteous and receptive morphisms of event structure with polarities is courteous and receptive; one can then see that isomorphisms of games are necessarily courteous and receptive. \square

Lemma 1.8. *Let $\sigma, \sigma' : A \rightarrow B$ and $\tau, \tau' : C \rightarrow D$ such that $\sigma \simeq \sigma'$ and $\tau \simeq \tau'$, then*

$$\sigma \otimes \tau \simeq \sigma' \otimes \tau'$$

Proof. This is similar to the congruence of \simeq with respect to \odot . For $\varphi_1 : S \cong S'$, $\varphi_2 : T \cong T'$, the isomorphisms given by $\sigma \simeq \sigma'$ and $\tau \simeq \tau'$, the above lemma is a consequence of

$$\begin{array}{ccc} \varphi : \mathcal{C}(S \otimes T) & \cong & \mathcal{C}(S' \otimes T') \\ x_S \otimes x_T & \mapsto & \varphi_1(x_S) \otimes \varphi_2(x_T) \end{array}$$

being an isomorphism, such that $\hat{\varphi}$ commutes with $\sigma \otimes \tau$ and $\sigma' \otimes \tau'$. \square

For bifunctionality, one can easily see that the tensor product preserves identities:

Proposition 1.6. *For every two games A and B , $\mathfrak{C}_{A \otimes B} \simeq \mathfrak{C}_A \otimes \mathfrak{C}_B$.*

Proof. It is direct to check that

$$\begin{array}{ccc} \mathfrak{C}_A \parallel \mathfrak{C}_B & \xrightarrow{\quad \gamma \quad} & \mathfrak{C}_{A \otimes B} \\ & \searrow \mathfrak{C}_A \otimes \mathfrak{C}_B & \swarrow \mathfrak{C}_{A \otimes B} \\ & A^\perp \parallel B^\perp \parallel A \parallel B & \end{array}$$

\square

The preservation of composition is a little more involved, we only sketch a proof for it and redirect the reader to [CCRW17] or chapter 3 for more details:

Proposition 1.7. *Let*

$$\begin{array}{ll} \sigma_1 : S_1 \rightarrow A_1^\perp \parallel B_1 & \tau_1 : T_1 \rightarrow B_1^\perp \parallel C_1 \\ \sigma_2 : S_2 \rightarrow A_2^\perp \parallel B_2 & \tau_2 : T_2 \rightarrow B_2^\perp \parallel C_2 \end{array}$$

be strategies, then,

$$(\tau_1 \odot \sigma_1) \otimes (\tau_2 \odot \sigma_2) \simeq (\tau_1 \otimes \tau_2) \odot (\sigma_1 \otimes \sigma_2)$$

Proof sketch. Based on the observation that interaction is a pullback, there is an isomorphism:

$$\begin{array}{ccc} (T_1 \otimes S_1) \parallel (T_2 \otimes S_2) & \xleftarrow{\sim} & (T_1 \parallel T_2) \otimes (S_1 \parallel S_2) \\ \downarrow (\tau_1 \otimes \sigma_2) \parallel (\tau_1 \otimes \sigma_2) & & \downarrow (\tau_1 \otimes \tau_2) \otimes (\sigma_1 \otimes \sigma_2) \\ (A_1 \parallel B_1 \parallel C_1) \parallel (A_2 \parallel B_2 \parallel C_2) & \xrightarrow{\sim} & (A_1 \parallel A_2) \parallel (B_1 \parallel B_2) \parallel (C_1 \parallel C_2) \end{array}$$

This isomorphism preserves visible events (*i.e.* those that map to A_i or C_i) hence it projects to an isomorphism between $(T_1 \odot S_1) \parallel (T_2 \odot S_2)$ and $(T_1 \parallel T_2) \odot (S_1 \parallel S_2)$. \square

Finally the symmetric monoidal structure $(\text{CG}, \otimes, \emptyset)$, is obtained as a *lifting* of the symmetric monoidal structure of $(\mathcal{E}, \parallel, \emptyset)$.

Based on lemma 1.7 a first step is to note that post composition can be used to rename the game components of a strategy:

Definition 1.14. Let $\sigma : A \rightarrow B$ be a strategy and $f : A \cong A'$ be a game isomorphism, the *left renaming* of σ via f is the strategy

$$(f \cdot \sigma) : S \rightarrow A' \parallel B \xrightarrow{f \parallel B} A' \parallel B$$

Similarly, for $g : B \cong B'$, the strategy $(\sigma \cdot g) : A \rightarrow B' = (A \parallel g) \circ \sigma$ defines the *right renaming* of σ via g .

When acting on the outer components, renaming preserve composition.

Proposition 1.8. Let $f : A \cong A'$, $g : C \cong C'$ be isomorphisms of games, let $\sigma : A \rightarrow B$, $\tau : B \rightarrow C$, be strategies then

$$\begin{aligned} \tau \odot (f \cdot \sigma) &\simeq f \cdot (\tau \odot \sigma) \\ (\tau \cdot g) \odot \sigma &\simeq (\tau \odot \sigma) \cdot g \end{aligned}$$

Proof sketch. Based on the observation that interaction is a pullback, there is an isomorphism:

$$\begin{array}{ccccc} & & T \otimes S & \xleftarrow{\sim} & T \otimes S & & \\ & \swarrow & \downarrow & \swarrow & \downarrow & \searrow & \\ S \parallel C & \xleftarrow{\sim} & A \parallel T & \xleftarrow{\sim} & S \parallel C & \xleftarrow{\sim} & A' \parallel T \\ \sigma \parallel C \searrow & & \swarrow A \parallel \tau & & \sigma \parallel C \downarrow & & \swarrow A' \parallel \tau \\ & & A \parallel B \parallel C & & A \parallel B \parallel C & & \\ & & \downarrow f \parallel B \parallel C & & \downarrow f \parallel B \parallel C & & \\ & & A' \parallel B \parallel C & & A' \parallel B \parallel C & & \end{array}$$

As above, this isomorphism preserves visible events and so defines an isomorphism for $\tau \odot (f \cdot \sigma) \simeq f \cdot (\tau \odot \sigma)$ after projection. \square

The above proposition is particularly interesting in the case of copycat renaming as it yields a functorial action:

Proposition 1.9. *Let $f : A \cong A'$ be an isomorphism of game, its lifting, \mathfrak{C}_f , is defined as the copycat right renaming*

$$\mathfrak{C}_f : A \twoheadrightarrow A' = \mathfrak{C}_A \cdot f$$

For $f : A \cong B$ and $g : B \cong C$ two isomorphisms of games, we have

$$\mathfrak{C}(g \circ f) \simeq (\mathfrak{C}_g) \odot (\mathfrak{C}_f)$$

Moreover, for every game A , $\mathfrak{C}_A \cdot id_A = \mathfrak{C}_A$

Using the above construction, the unitors, associators and commutators of $(\mathcal{E}, \parallel, \emptyset)$ can thus be lifted to isomorphic strategies in CG:

$$\begin{array}{lcl} \rho_A & : & A \parallel \emptyset \rightarrow A \\ \lambda_A & : & \emptyset \parallel A \rightarrow A \\ s_{A,B} & : & A \parallel B \rightarrow B \parallel A \\ \alpha_{A,B,C} & : & (A \parallel B) \parallel C \rightarrow A \parallel (B \parallel C) \\ & \rightsquigarrow & \\ & & \mathfrak{C}_{\rho_A} : A \otimes \mathbf{1} \twoheadrightarrow A \\ & & \mathfrak{C}_{\lambda_A} : \mathbf{1} \otimes A \twoheadrightarrow A \\ & & \mathfrak{C}_{s_{A,B}} : A \otimes B \twoheadrightarrow B \otimes A \\ & & \mathfrak{C}_{\alpha_{A,B,C}} : (A \otimes B) \otimes C \twoheadrightarrow A \otimes (B \otimes C) \end{array}$$

By the functorial properties of the lifting, all the commuting diagram satisfy by these isomorphisms in \mathcal{EP} are lifted to CG. Furthermore, one can check that these isomorphisms are natural, hence defining the unitors, associators, and commutators for $(CG, \otimes, \mathbf{1})$. This leads to the conclusion that:

Theorem 1.2. *Games and strategies up to isomorphism, together with \odot , $(\mathbf{1}, \otimes)$ and $(_)\perp$ form a compact closed category CG.*

PART I

Annotated strategies

Annotated strategies

This part presents the core framework of this thesis: an enrichment of the concurrent games model based on event structures [CCRW17] that allows for *annotated* strategies. Annotated strategies correspond to regular plain strategies where every Player move is played together with some *side-information* whose value may vary according to those received from Opponent moves. These pieces of information are said to be “on the side” as they do not interfere with the interactions/execution flow of the strategies, meaning that composing two annotated strategies preserves their causal plain structure.

The construction presented here is a unified model for the semantics developed in the next two parts of this thesis, it thus encompasses the case where those pieces of side-information are terms that represent *witnesses* in valid first-order classical proofs, and the case where they are functions over reals that help *analyse the resource consumption* of higher order concurrent programs with shared memory. More generally, this framework allows for annotations with morphisms from any *cartesian category* and can also be refined in order to reflect additional structure on the annotations such as *ordering*.

In this part, we also show that the games model enriched with annotated strategies – that are enrichment of CG – can be simplified to enrich other concurrent games models such as concurrent games *without non-determinism* or concurrent games and *rigid* strategies, that are strategies validating the idempotence of non-determinism [CC16] (*i.e.* $a \vee a = a$). These constructions follow the correspondence between CG and these simpler models, which we will recall. These simplified models are of interest as they will be the actual models used in the next two parts.

The proofs techniques used in this development are those recalled in chapter 1, inspired by [CCRW17]. Our main technical contribution lies in the definition of a category of *event structures with annotations* that admits pullbacks. In particular, we believe that this framework could serve as a basis for other games models of annotated strategies in which annotations *do influence* interactions of strategies. This direction however falls out of the scope of this thesis.

Outline Chapter 2 presents our general construction for concurrent games model with annotated strategies. It defines $\mathbb{T}\text{-CG}$, the parametric category of games and strategies with annotations as terms from an *(in)equational theory* \mathbb{T} . In particular, we show that both of the application cases from part II and III mentioned above can be expressed as $\mathbb{T}\text{-CG}$ for a well chosen \mathbb{T} .

Chapter 3 provides a detailed proof of the categorical structure of \mathbb{T} -CG – a compact closed category. The focus is actually put on Σ -CG, an instance of \mathbb{T} -CG whose categorical structure is equivalent to the general \mathbb{T} -CG after quotient. This proof is widely inspired from the proof of the categorical structure of CG found in [CCRW17]. Its main novelty is the description of a category of *annotated* event structures in which annotations may interfere with interaction.

Chapter 4 concludes this part with a description of the two simplified model of \mathbb{T} -CG actually used in the applicative cases of part II and III. As for CG, we first recall the construction of these models in the plain case, before enriching them with annotations.

Concurrent games with annotations

This chapter presents a general construction for extending the category CG of concurrent games and strategies (as recalled in chapter 1) with annotations on strategies. It starts with a simple enrichment, where annotations correspond to terms over a signature. This is then generalised to annotations with expressions from an (in)equational theory, by the means of equivalence classes of terms and order upon them. To illustrate the expressiveness of this latest construction, a series of examples of instantiation is provided, generalising further the setting to multi-sorted equational theory. In particular this yields to a model of concurrent games and strategies with annotations as morphisms from a cartesian category.

2.1 Σ -strategies

2.1.1 Preliminaries on terms

First order terms and signatures. A first order *signature* Σ is a set of symbol declarations $s \in \Sigma^n$, where n is the *arity* of s , that is, its number of arguments (possibly null). Given a signature Σ and a set of variables \mathcal{V} (disjoint from Σ), the set of *terms over* \mathcal{V} generated by Σ is inductively defined by:

$$\begin{array}{l} t_1, \dots, t_n \in \text{Tm}_\Sigma(\mathcal{V}) \quad := \quad x \in \mathcal{V} \\ \quad \quad \quad \quad \quad \quad \quad | \quad (s) t_1 \dots t_n \quad \text{with } s \in \Sigma^n \end{array}$$

For t a term, its set of *free variables* is denoted $\text{fv}(t) \subseteq \mathcal{V}$. By extension, if T is a set of terms then $\text{fv}(T) = \bigcup_{t \in T} \text{fv}(t)$. A term is *closed* if it does not contain any free variables.

The category Subst_Σ . Given two sets \mathcal{V} and \mathcal{V}' and a signature Σ , a Σ -*substitution* is a mapping $\rho : \mathcal{V} \xrightarrow{\Sigma} \mathcal{V}'$ that associates every variable $x \in \mathcal{V}$

with a term $\rho(x) \in \text{Tm}_\Sigma(\mathcal{V})$. Substitutions are described by their graphs $[x \mapsto \rho(x)]_{x \in \mathcal{V}'}$, shortened $[\rho]$.

The *substitution of a term t by ρ* , denoted $t[\rho]$, simultaneously replaces every \mathcal{V}' -variable in t by its image via ρ . It is inductively defined on terms:

$$\begin{cases} v[\rho] = v & \text{if } v \notin \mathcal{V}' \\ v[\rho] = \rho(v) & \text{if } v \in \mathcal{V}' \\ ((s) t_1 \dots t_n)[\rho] = (s) t_1[\rho] \dots t_n[\rho] \end{cases}$$

Substitutions can be *composed*

Definition 2.1. Let $\rho : \mathcal{V}_1 \xrightarrow{\Sigma} \mathcal{V}_2$ and $\nu : \mathcal{V}_2 \xrightarrow{\Sigma} \mathcal{V}_3$ be two substitutions, then $\nu \circ \rho : \mathcal{V}_1 \xrightarrow{\Sigma} \mathcal{V}_3$ is defined as

$$\begin{aligned} \nu[\rho] : \mathcal{V}_3 &\rightarrow \text{Tm}_\Sigma(\mathcal{V}_1) \\ x &\mapsto \nu(x)[\rho] \end{aligned}$$

Composition is associative and has obvious identities. Moreover for any two $\mathcal{V}_1, \mathcal{V}_2$,

$$\begin{array}{ccc} & \mathcal{V}_1 \parallel \mathcal{V}_2 & \\ \iota_1 \swarrow & & \searrow \iota_2 \\ \mathcal{V}_1 & & \mathcal{V}_2 \end{array}$$

defines a cartesian product – for \parallel the disjoint union $\mathcal{V}_1 \uplus \mathcal{V}_2$ and ι_i the corresponding injections in Set/projections in Subst. Hence,

Theorem 2.1. *Given a signature Σ , sets and Σ -substitutions over them form a cartesian category, denoted Subst_Σ .*

Transposition. Given two isomorphic sets of variables $\rho : \mathcal{V}_1 \cong \mathcal{V}_2$, the bijection ρ defines an isomorphism in Subst_Σ that can be used to *transport* substitutions. For, $\mu : \mathcal{V}_1 \xrightarrow{\Sigma} \mathcal{V}_1$ we define its *transport via ρ* , by

$$\rho.\mu : \mathcal{V}_2 \xrightarrow{\Sigma} \mathcal{V}_2 = \rho^{-1} \circ \mu \circ \rho$$

In particular $_[\mu][\rho] = _[\rho][\rho.\mu]$, $\rho^{-1}.\rho.\mu = \mu$, and for $\rho_1 : \mathcal{V}_1 \cong \mathcal{V}_2$, $\rho_2 : \mathcal{V}_2 \cong \mathcal{V}_3$, $\rho_2.\rho_1.\mu = (\rho_2 \circ \rho_1).\mu$.

Syntactic unification. Given two terms $t \in \text{Tm}_\Sigma(\mathcal{V}_1), t' \in \text{Tm}_\Sigma(\mathcal{V}_2)$, we say that t' is an *instance* of t , written $t \preceq t'$, if there is a substitution $\rho : \mathcal{V}_2 \xrightarrow{\Sigma} \mathcal{V}_1$ such that $t[\rho] = t'$. This defines a pre-order on terms which extends to substitutions as follows:

Definition 2.2. Let $\nu : \mathcal{V}_1 \xrightarrow{\Sigma} \mathcal{V}_0$, $\rho : \mathcal{V}_2 \xrightarrow{\Sigma} \mathcal{V}_0$, we say that ν *subsumes* ρ , written $\nu \preceq \rho$, if there is a substitution $\mu : \mathcal{V}_2 \xrightarrow{\Sigma} \mathcal{V}_1$ such that $\nu \circ \mu = \rho$. (Equality here is the usual component-wise equality.)

In unification theory, this can be rephrased as a *matching problem* [Gog89]: saying that $\nu \preceq \rho$ is equivalent to saying that there exists a solution $\mu : \mathcal{V}_2 \xrightarrow{\Sigma} \mathcal{V}_1$ to the finite set of inequations

$$\{\nu(x) \doteq \rho(x)\}_{x \in \mathcal{V}_0}$$

In general, a matching problem does not have a unique solution.

Example 2.1. Consider $\nu, \rho : \{x_1, x_2\} \xrightarrow{\Sigma} \{x_1, x_2\}$ two parallel substitutions such that $\nu = [(f)x_2/x_1, x_2/x_2]$, $\rho = [(f)c/x_1, x_2/x_2]$, then any substitution $\mu = [t/x_1, c/x_2]$ for $t \in \text{Tm}_{\Sigma}(\{x_1, x_2\})$ is a solution to the corresponding matching problem. In particular, ρ itself is a solution.

In the example above, one can note that ν and ρ are *idempotent* substitutions, meaning that they are endo-substitutions $\nu : \mathcal{V} \rightarrow \mathcal{V}$ such that $\nu \circ \nu = \nu$. In the particular case of idempotent substitution the above remark on ρ being a solution to the matching problem is always true:

Lemma 2.1. *Let $\nu : \mathcal{V} \xrightarrow{\Sigma} \mathcal{V}$, $\rho : \mathcal{V}' \xrightarrow{\Sigma} \mathcal{V}$ be two substitutions such that $\nu \preceq \rho$ and ν is idempotent, then $\nu \circ \rho = \rho$.*

Proof. By definition there exists $\mu : \mathcal{V}' \xrightarrow{\Sigma} \mathcal{V}$ such that $\nu \circ \mu = \rho$, hence, by idempotence $\rho = \nu \circ \mu = (\nu \circ \nu) \circ \mu = \nu \circ (\nu \circ \rho)$. This is equivalently shown on the following diagram. \square

$$\begin{array}{ccc} \mathcal{V}' & \xrightarrow{\rho} & \mathcal{V} \\ \downarrow \mu & \nearrow \nu & \uparrow \nu \\ & \mathcal{V} & \\ \downarrow \nu & \searrow \nu & \downarrow \nu \\ & \mathcal{V} & \\ \downarrow \rho & \xrightarrow{\nu} & \mathcal{V} \end{array}$$

In unification theory, we also say that two terms $t, t' \in \text{Tm}_{\Sigma}(\mathcal{V})$ can be *equalised* if there is a substitution $\mu : \mathcal{V} \xrightarrow{\Sigma} \mathcal{V}$ such that $t_1[\mu] = t_2[\mu]$. *Unification problems*, then, are finite sets of term-equations of the form

$$\{t_i \doteq t'_i\}_{i \in I}$$

for some index set I . In Subst_{Σ} they are captured via the notion of equalizer [Gog89]:

Theorem 2.2 (“Herbrand-Robinson”). *Two parallel substitutions $\rho, \nu : \mathcal{V}_1 \xrightarrow{\Sigma} \mathcal{V}_0$ have an equalizer $\mu : \mathcal{V}_2 \xrightarrow{\Sigma} \mathcal{V}_1$ iff the unification problem $\mathcal{U}_{\rho, \nu} = \{\rho(v) \doteq \nu(v)\}_{v \in \mathcal{V}_0}$ has a solution.*

Sketch. That μ is a solution for the corresponding unification problem is immediate since $\rho \circ \mu = \nu \circ \mu$ by definition.

The contrapositive arises from the actual Herbrand-Robinson theorem that states that if a matching problem has a solution then it has one that is minimal for \preceq , called its *most general unifier* [BN98]. Let \mathcal{U} be a unification problem with free variables in \mathcal{V} , an algorithm for computing this most general unifier is as follows:

```

let rec mgu  $\mathcal{U}$  = match  $\mathcal{U}$  with
|  $\emptyset \rightarrow \text{id}_{\mathcal{V}}$ 
|  $\mathcal{U} \uplus \{v \doteq t\}$  or  $\mathcal{U}' \uplus \{t \doteq v\}$  with  $v \in \mathcal{V}$  and  $v \notin \text{fv}(t)$ 
   $\rightarrow$  let  $\mu = \mathbf{mgu} \mathcal{U}[v \mapsto t]$  in  $\mu[v \mapsto t[\mu]]$ 
|  $\mathcal{U} \uplus \{(f) s_1 \dots s_n \doteq (f) t_1 \dots t_n\}$  with  $f \in \Sigma^n$ 
   $\rightarrow$  let  $\mu' = \mathbf{mgu} \mathcal{U} \uplus \{s_i \doteq t_i\}_{i=1}^n$  in  $\mu[\mu']$ 
|  $\_ \rightarrow \text{Abort}$ 

```

Using an appropriate measure it is easy to show that this algorithm terminates and one can follow the same induction to show that the return value has the universal property of an equalizer. \square

The algorithm above will be crucial in the process of composition of strategies with term annotations introduced later. Taking ρ and ν as in example 2.1, a possible run of the algorithm is

$$\begin{aligned}
(\mathcal{U}, \mu) &\doteq (\{(f)x_2 \doteq (f)c, x_2 \doteq x_2\}, [x_1/x_1, x_2/x_2]) \\
&\rightarrow (\{x_2 \doteq c, x_2 \doteq x_2\}, [x_1/x_1, x_2/x_2]) \\
&\quad \rightarrow (\{c \doteq c\}, [x_1/x_1, c/x_2]) \\
&\quad \rightarrow (\{\}, [x_1/x_1, c/x_2])
\end{aligned}$$

Because ν and ρ share the same set of variable, an other unification problem can be issued:

$$\mathcal{U}'_{\rho, \nu} = \{v \doteq \rho(v), v \doteq \nu(v)\}_{v \in \mathcal{V}}$$

In that case, a possible run of the algorithm is

$$\begin{aligned}
(\mathcal{U}, \mu) &\doteq (\{x_1 \doteq (f)x_2, x_1 \doteq (f)c, x_2 \doteq x_2\}, [x_1/x_1, x_2/x_2]) \\
&\rightarrow (\{(f)x_2 \doteq (f)c, x_2 \doteq x_2\}, [(f)x_2/x_1, x_2/x_2]) \\
&\quad \rightarrow (\{x_2 \doteq c, x_2 \doteq x_2\}, [(f)x_2/x_1, x_2, x_2]) \\
&\quad \quad \rightarrow (\{c \doteq c\}, [(f)c/x_1, c/x_2]) \\
&\quad \quad \rightarrow (\{\}, [(f)c/x_1, c/x_2])
\end{aligned}$$

One can note that in the above the resulting substitution μ is idempotent and such that $\rho \preceq \mu$ and $\nu \preceq \mu$. In fact, reasoning again by induction on the algorithm one can show that:

Corollary 2.1. *Let $\rho, \nu : \mathcal{V} \xrightarrow{\Sigma} \mathcal{V}$ be two idempotent parallel substitutions, if the unification problem $\mathcal{U}'_{\rho, \nu}$ has a solution then ρ and ν have an idempotent maximum with respect to \preceq :*

$$\rho \preceq \mu \succeq \nu$$

Moreover, by lemma 2.1, $\mu : \mathcal{V} \xrightarrow{\Sigma} \mathcal{V}$ is such that $\mu \circ \rho = \mu = \mu \circ \nu$.

Furthermore, one can make explicit the relation between the unification problems $\mathcal{U}_{\rho, \nu}$ and $\mathcal{U}'_{\rho, \nu}$:

Lemma 2.2. *Let $\rho, \nu : \mathcal{V} \xrightarrow{\Sigma} \mathcal{V}$ be two idempotent parallel substitutions, then $\mathcal{U}_{\rho, \nu}$ is solvable iff $\mathcal{U}'_{\rho, \nu}$ is solvable.*

Moreover, every solution for $\mathcal{U}'_{\rho, \nu}$ is a solution for $\mathcal{U}_{\rho, \nu}$.

Proof. The left implication follows by definition: if μ is a solution for $\mathcal{U}'_{\rho, \nu}$, then for every $v \in \mathcal{V}$, $\rho(v)[\mu] = v[\mu] = \nu(v)[\mu]$ hence μ is a solution for $\mathcal{U}_{\rho, \nu}$.

For the right implication, consider μ a solution of $\mathcal{U}_{\rho, \nu}$ and take $\mu' : \mathcal{V} \xrightarrow{\Sigma} \mathcal{V}$ to be the substitution equal to μ on $F = \text{fv}(\rho(\mathcal{V})) \cup \text{fv}(\nu(\mathcal{V}))$ but defined by $\mu'(v) = \nu(v)[\mu]$ for every $v \notin F$. By definition, μ' is a solution for $\mathcal{U}_{\rho, \nu}$: every change that have been made in comparison with μ are on variables that do not appear in $\mathcal{U}_{\rho, \nu}$. We are now in a good shape to prove that μ' also defines a solution for $\mathcal{U}'_{\rho, \nu}$: let us first note that if $v \in F$ then, by idempotence, either $\rho(v) = v$ or $\nu(v) = v$, so $\rho(v)[\mu'] = v[\mu'] = \nu(v)[\mu']$; now, if $v \notin F$ then, by definition $v[\mu'] = \rho(v)[\mu'] = \nu(v)[\mu']$. Concluding that $\mathcal{U}'_{\rho, \nu}$ is solvable. \square

This algorithmic and categorical view will both give insight to further development, especially in Chapter 3.

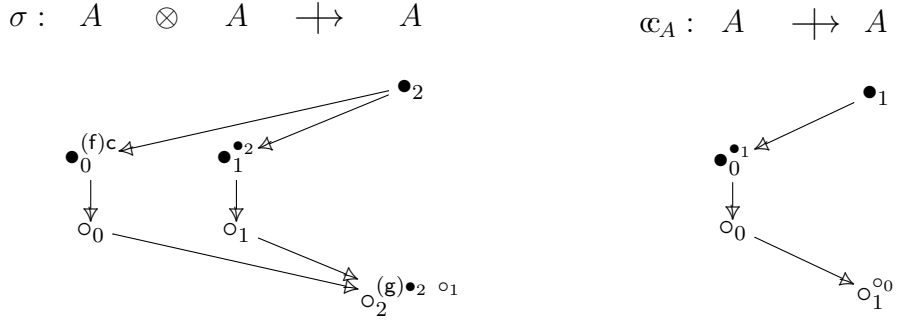
2.1.2 Σ -strategies

We now give a first definition of annotated strategies, restricted to annotations with terms over a signature. The aim is to define a game model where every event is played together with some information – in the shape of a first order term. To this end, strategies provide for every positive move an open term, its *annotation*, that describes how this information relates to the one of its negative predecessors. This follows the idea that a strategy is defined in reaction to its opponent behaviour.

Definition 2.3. Let Σ be a first order signature, a Σ -strategy is the data of a plain strategy $\sigma : S \rightarrow A$ together with an Σ -annotation

$$\lambda_{\sigma} : (s \in S^+) \rightarrow \text{Tm}_{\Sigma}([s]^-)$$

We write $\sigma : A \xrightarrow{\Sigma} B$ for Σ -strategies from A to B .

Figure 2.1: Examples of Σ -strategies

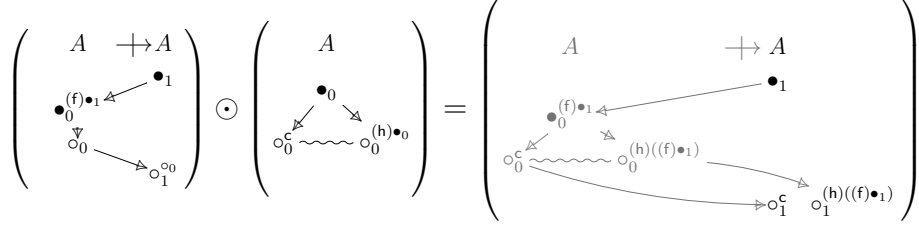
The above definition says that every positive move of a Σ -strategy comes equipped with a term whose free variables are named after the negative moves in its causal history. These variables can be thought as unknown data. In a game model, it makes sense for these data to be provided by Opponent via its own annotations. The restriction to negative variables in the causal history of a positive move is there to ensure that the data needed is always provided by the time a move is actually played. Other than that, there are no restrictions on how to use predecessors' informations. This is illustrated in Figure 2.1 where positive events are decorated by their annotations in superscript. On the left terms are built over $\Sigma \supseteq \{c^0, f^1, g^2\}$ and $\lambda_\sigma(\bullet_0)$, $\lambda_\sigma(\bullet_1)$, $\lambda_\sigma(\circ_2)$ respectively depend on *no* variables, *all* negative predecessors (here this also matches the set of all immediate predecessors) and *some* negative predecessors. On the right, we give the annotated version of the copycat strategy for A which we will comment later on.

As in the plain case, the actual names of events in the support of a Σ -strategy do not matter; only their structure and annotations count. Following definition 1.8, Σ -strategies are considered to be isomorphic up to renaming of their events:

Definition 2.4. Let σ, σ' be two Σ -strategies on the same game, then σ and σ' are *isomorphic*, denoted $\sigma \simeq \sigma'$, iff $\sigma \stackrel{\mathcal{L}}{\simeq} \sigma'$ as plain strategies and φ preserves annotations *i.e.* $\varphi.\lambda_\sigma = \lambda_{\sigma'}$.

This definition will find another justification later on (via definition 3.10 and proposition 3.9).

Composition. Σ -strategies compose following the same interaction and hiding process as plain strategies (see section 1.2.2 and 1.2.3) but with additional work to compose their annotations. Annotations on their interaction

Figure 2.2: Example of interaction and composition between two Σ -strategies

are defined by the mean of (mutually inductive) substitutions: Every negative free variable in the annotation of a strategy that is associated with a move from the common game is substituted by the annotation of its corresponding move in the counter-strategy. If however these negative variables correspond to a move in the outer games, then it is left open. More formally:

Lemma 2.3. *Let $\sigma : A \xrightarrow{\Sigma} B$ and $\tau : B \xrightarrow{\Sigma} C$ be two Σ -strategies then there is a unique*

$$\lambda_{\tau \circledast \sigma} : T \circledast S \rightarrow \text{Tm}_{\Sigma}(T \circledast S)$$

such that $\lambda_{\tau \circledast \sigma}(\llbracket s, t \rrbracket_{\varphi}) =$

1. $\lambda_{\sigma}(s)[\Pi_{1,\varphi}^{-1}][\lambda_{\tau \circledast \sigma}]$ if $s \in S^+$,
2. $\lambda_{\tau}(t)[\Pi_{2,\varphi}^{-1}][\lambda_{\tau \circledast \sigma}]$ if $t \in T^+$,
3. $\llbracket s, t \rrbracket_{\varphi}$ otherwise.

Proof. The function $\lambda_{\tau \circledast \sigma}$ can be defined by well-founded induction on $<_{\tau \circledast \sigma}$ following the above characterisation: cases are disjoint and exhaustive, and for the first two cases, the substitution by $\lambda_{\tau \circledast \sigma}$ is well defined since free variables in $\lambda_{\sigma}(s)$ are negative predecessors of s in σ and so have a corresponding event in the causal history of $\llbracket s, t \rrbracket_{\varphi}$. That is

$$\text{fv}(\Pi_{1,\varphi}^{-1} \circ \lambda_{\sigma}(s)) \subseteq \Pi_{1,\varphi}^{-1}([s]^{-}) \subseteq \llbracket s, t \rrbracket_{\varphi} T \circledast S$$

And similarly for $\Pi_{2,\varphi}^{-1} \circ \lambda_{\tau}(t)$. \square

Figure 8.9 displays an example of interaction between two Σ -strategies. The resulting composition is obtained as in the plain case by hiding away the grey events, that are moves from the shared game A on the left. Hiding in this case is still well-defined at the level of annotations as for every $\llbracket s, t \rrbracket_{\varphi}$

$$\text{fv}(\lambda_{\tau \circledast \sigma}(\llbracket s, t \rrbracket_{\varphi})) \subseteq \varphi^{-}$$

where φ^- is a shortcut to designate events in the causal history of $\llbracket s, t \rrbracket_\varphi$ whose projection via $\tau \circledast \sigma$ is negative in $A^\perp \parallel C$. We set

Definition 2.5. The canonical annotation for $\tau \odot \sigma$ is defined by

$$\lambda_{\tau \odot \sigma} := \lambda_{\tau \circledast \sigma | (T \odot S)^+}$$

Categorical structure. The compact closed structure of CG is preserved by its extension with annotations. Let's first focus on the purely categorical part (forgetting for now the compact closure).

For associativity of composition one can simply check that given $\sigma : A \xrightarrow{\Sigma} B$, $\tau : B \xrightarrow{\Sigma} C$ and $\rho : C \xrightarrow{\Sigma} D$ the isomorphism $\varphi : \rho \odot (\tau \odot \sigma) \simeq (\rho \odot \tau) \odot \sigma$ defined at the level of plain strategies (cf Proposition 1.3) preserves annotations. However, this requires to make explicit the above isomorphism and a fastidious check. In Section 3.1.2, we'll see that annotations of interactions actually correspond to the most general unifiers of some unification problems and prove associativity from there using their universal property. For now we simply claim:

Proposition 2.1. *Let $\sigma : A \xrightarrow{\Sigma} B$, $\tau : B \xrightarrow{\Sigma} C$, $\rho : C \xrightarrow{\Sigma} D$ with isomorphism $\varphi : \rho \odot (\tau \odot \sigma) \simeq (\rho \odot \tau) \odot \sigma$ on plain strategies given by proposition 1.3, then $\varphi \cdot \lambda_{\rho \odot (\tau \odot \sigma)} = \lambda_{(\rho \odot \tau) \odot \sigma}$.*

Figure 2.1 shows how plain copycat strategies can be given an annotated version: annotations on positive moves simply forwards the annotation received from their negative immediate predecessors.

Definition 2.6. Let A be a game, $\mathfrak{c}_A : A \xrightarrow{\Sigma} A$ is the plain copycat strategy with annotation

$$\lambda_{\mathfrak{c}_A}((i, a)^+) = (1 - i, a)^-$$

As expected, this strategy is still neutral for composition.

Proposition 2.2. *For every Σ -strategies $\sigma : A \xrightarrow{\Sigma} B$, $\tau : C \xrightarrow{\Sigma} A$,*

$$\sigma \odot \mathfrak{c}_A \simeq \sigma \quad \text{and} \quad \mathfrak{c}_A \odot \tau \simeq \tau$$

Proof. We focus on the case where $\sigma : A$ is post-composed by \mathfrak{c}_A ; pre-composition is symmetric and the more general case where $\sigma : B \xrightarrow{\Sigma} A$ follows the same reasoning, only with more technical details.

Building up on the proof for plain strategies 1.5, there is an isomorphism $\chi : S \cong \mathbb{C}_A \odot S$ that sends $s \in S$ to the prime and visible event $(\sigma([s]^*) \parallel$

$\sigma[s]) \odot [s]^* \in \mathbb{C}_A \odot S$. Hence to conclude on the above it is enough to check that this isomorphism preserves annotations.

Let φ be the prime secured bijection defined by $(\sigma([s]^*) \parallel \sigma[s]) \odot [s]^*$ and let x be the corresponding configuration in $\mathbb{C}_A \otimes S$. Following the equivalence between secured bijections and configurations in the interaction of two strategies (lemma 1.5), the annotations on $\mathbb{C}_A \otimes S$ can be transported onto synchronised pairs in φ to define

$$\lambda_\varphi = \nu_x.(\lambda_{\mathbb{C}_A \otimes S})|_x : \varphi \rightarrow \text{Tm}_\Sigma(\varphi)$$

– recall that $\nu_x : x \simeq \varphi$. Writing $(s, \sigma(s))$ and (a, a) respectively for the synchronised pairs of events $((0, s), (0, \sigma(s)))$ and $((1, a), (1, a))$ in φ , lemma 2.3 simplifies into

1. $\lambda_\varphi(\sigma(s^-), \sigma(s^-)) = (\sigma(s^-), \sigma(s^-)) = (\nu_x \circ \chi)(s^-)$
2. Hence,

$$\begin{aligned} \lambda_\varphi(s^-, \sigma(s^-)) &= \lambda_{\mathbb{C}_A}(0, \sigma(s^-))[\pi_2^{-1}][\lambda_\varphi] \\ &= (1, \sigma(s^-))[\pi_2^{-1}][\lambda_\varphi] \\ &= (\sigma(s^-), \sigma(s^-))[\lambda_\varphi] \\ &= (\sigma(s^-), \sigma(s^-))[\chi][\nu_x] \quad (\text{by 1.}) \end{aligned}$$
3. Hence, $\lambda_\varphi(s^+, \sigma(s^+)) = \lambda_\sigma(s^+)[\pi_1^{-1}][\lambda_\varphi] = \lambda_\sigma(s^+)[\chi][\nu_x]$ since by definition $\text{fv}(\lambda_\sigma(s^+)) \subseteq [s^+]^-$ and so $\text{fv}(\lambda_\sigma(s^+))[\pi_1^{-1}] \subseteq \bigcup_{s' \in [s^+]^-} (s', \sigma(s'))$
4. Finally,

$$\begin{aligned} \lambda_\varphi(\sigma(s^+), \sigma(s^+)) &= \lambda_{\mathbb{C}_A}(1, \sigma(s^+))[\pi_2^{-1}][\lambda_\varphi] \\ &= (0, \sigma(s^+))[\pi_2^{-1}][\lambda_\varphi] \\ &= \lambda_\varphi(s^+, \sigma(s^+)) \\ &= \lambda_\sigma(s^+)[\chi][\nu_x] \quad (\text{by 3.}) \end{aligned}$$

Hence for every $s \in S^+$, $\lambda_{\mathbb{C}_A \odot \sigma}(\chi(s)) = \lambda_\sigma(s)$. \square

The compact closed structure of CG also extends smoothly to Σ -strategies. Recall that on plain games and strategies, tensoring only acts as a renaming: For $\sigma : S \rightarrow A^\perp \parallel B$ and $\tau : T \rightarrow C^\perp \parallel D$,

$$\sigma \otimes \tau = \gamma \circ (\sigma \parallel \tau) : A \otimes C \rightarrow B \otimes D$$

where $\gamma : (A^\perp \parallel A) \parallel (B^\perp \parallel B) \rightarrow (A^\perp \parallel B^\perp) \parallel (A \parallel B)$. This renaming has no impact on the support, $S \parallel T$, of $\sigma \parallel \tau$, and it is natural to take the product $\lambda_\sigma \parallel \lambda_\tau$ as annotation:

$$\begin{aligned} \lambda_{\sigma \otimes \tau} = (\lambda_\sigma \parallel \lambda_\tau) : (e \in (S \parallel T)^+) &\rightarrow \text{Tm}_\sigma([e]_{S \parallel T}^-) \\ (0, s) &\mapsto \lambda_\sigma(s)[t_0] \\ (1, t) &\mapsto \lambda_\tau(t)[t_1] \end{aligned}$$

One can check that in particular

Proposition 2.3. *There is an isomorphism $\mathfrak{C}_{A \otimes B} \simeq \mathfrak{C}_A \otimes \mathfrak{C}_B$.*

Proof. By Proposition 1.6 the renaming γ defines an isomorphism from $\mathfrak{C}_A \otimes \mathfrak{C}_B$ to $\mathfrak{C}_{A \otimes B}$ as plain strategies. This immediately lifts to annotations by unfolding the definition. For example, let $a \in A^+$, then $\lambda_{\mathfrak{C}_A \otimes \mathfrak{C}_B}(0, (1, a))[\gamma] = (0, (0, a))[\gamma] = (0, (0, a)) = \lambda_{\mathfrak{C}_{A \otimes B}}(1, (0, a)) = \lambda_{\mathfrak{C}_{A \otimes B}}(\gamma(0, (1, a)))$. \square

For compatibility of the tensor product \otimes with composition, one could also check that the isomorphism from proposition 1.7 preserves annotations. For $\sigma_1 : A_1 \xrightarrow{\Sigma} B_1$, $\tau_1 : B_1 \xrightarrow{\Sigma} C_1$, $\sigma_2 : A_2 \xrightarrow{\Sigma} B_2$, $\tau_2 : B_2 \xrightarrow{\Sigma} C_2$, we expect

$$(\tau_1 \odot \sigma_1) \otimes (\tau_2 \odot \sigma_2) \simeq (\tau_1 \otimes \tau_2) \odot (\sigma_1 \otimes \sigma_2)$$

to behave well. However this would again require to make the above isomorphism explicit and a fastidious check on annotations. As for associativity we leave the proof for chapter 3.

As intuited in the proof of proposition 2.3, the notion of strategy renaming extends straightforwardly to Σ -strategies:

Definition 2.7. Let $f : A \cong A'$ be a game isomorphism and $\sigma : A \xrightarrow{\Sigma} B$, then $f \cdot \sigma : A' \xrightarrow{\Sigma} B$ together with annotation λ_σ defines the *left renaming* of σ via f , also denoted $f \cdot \sigma : A' \xrightarrow{\Sigma} B$.

For $g : B \cong B'$, the *right renaming* $\sigma \cdot g : A \xrightarrow{\Sigma} B'$ is defined similarly.

Following section 1.3 where the symmetric monoidal structural strategies $\mathfrak{C}_\lambda, \mathfrak{C}_\rho, \mathfrak{C}_\alpha, \mathfrak{C}_s$ of CG are obtained by renaming of base copycat strategies via the corresponding structural morphisms in \mathcal{E} , we use the above definition to design in turn the corresponding Σ -strategies. The unit and co-unit η and ε are also defined as in section 1.3, respectively by post-composing annotated copycat strategies (viewed as event structure morphisms) with the isomorphisms $\lambda_{A^\perp \parallel A}^{-1}$ and $\rho_{A^\perp \parallel A}^{-1}$ from \mathcal{E} . Saving the congruence proof of \simeq with respect to \odot and \otimes for the next section (Proposition 2.5 and 2.6) and other details for Chapter 3, we simply claim that the compact closed structure of CG is preserved by the addition of Σ -annotations:

Theorem 2.3. *Games and Σ -strategies up to isomorphism, together with \odot , $(1, \otimes)$ and $(_)^\perp$ form a compact closed category Σ -CG.*

2.2 \mathbb{T} -strategies

Previous section gives an account of how plain concurrent strategies based on event structures can be extended with Σ -annotations, that are open terms associated to their positive events, such that these terms may vary when composing two strategies together.

These annotations were originally designed to capture existential witnesses in the semantics of first-order proofs. Yet, from a game semantics perspective, they can simply be viewed as side-information on strategies. It is then natural to ask for more general information than first order terms. We now describe how first order term annotations can be turned into annotations from the free algebra of an (in)equational theory (to be defined) such that the corresponding equality and/or partial order also reflect on strategies.

2.2.1 \mathbb{T} -algebras

Inspired by free-algebras for equational theories, we introduce here a notion of inequational theories and free-algebras for them. This construction is a base for having more general annotations than mere terms.

Free algebras. Given a first order signature Σ , an *algebra* for Σ is a pair $\mathcal{X} = (X, \llbracket - \rrbracket_X)$ consisting of a carrier set X together with interpretation functions $\llbracket \mathbf{s} \rrbracket_X : X^n \rightarrow X$ for each $\mathbf{s} \in \Sigma^n$. By induction this defines a function $\llbracket t \rrbracket_X : X^\mathcal{V} \rightarrow X$ for every $t \in \text{Tm}_\Sigma(\mathcal{V})$.

Algebras for Σ define a category $\Sigma\text{-Alg}$ where morphisms are functions $f : X \rightarrow Y$ on carrier sets such that for every $s \in \Sigma^n$, $\llbracket \mathbf{s} \rrbracket_X(f(x_1) \dots f(x_n)) = f(\llbracket \mathbf{s} \rrbracket_Y x_1 \dots x_n)$. There is an obvious forgetful functor $\mathcal{U} : \Sigma\text{-Alg} \rightarrow \text{Set}$ that retains carrier sets. Given a set \mathcal{V} , a Σ -algebra over \mathcal{V} , also denoted (\mathcal{V}, Σ) -algebra, is a set morphism $h : \mathcal{V} \rightarrow \mathcal{U}(\mathcal{A})$ for $\mathcal{A} \in \Sigma\text{-Alg}$.

Lemma 2.4. *Given a set \mathcal{V} , the identity-on-variable function $\text{id} : \mathcal{V} \rightarrow \text{Tm}_\Sigma(\mathcal{V})$ is the free Σ -algebra over \mathcal{V} – with obvious interpretation functions – simply written $\text{Tm}_\Sigma(\mathcal{V})$.*

Freeness is a universal property on object similar to the universal property of the left adjoint of a forgetful functor. For Σ -algebra, it means that any function of the form $h : \mathcal{V} \rightarrow \mathcal{U}(\mathcal{A})$ can be lifted to Σ -algebras through a unique morphism $h^* : \text{Tm}_\Sigma(\mathcal{V}) \rightarrow \mathcal{A}$ such that

$$\begin{array}{ccc} \text{Tm}_\Sigma(\mathcal{V}) & \xrightarrow{h^*} & \mathcal{U}(\mathcal{A}) \\ & \swarrow \text{id} & \nearrow h \\ & \mathcal{V} & \end{array}$$

commutes in Set . Here h^* is simply defined by $\llbracket - \rrbracket_{\mathcal{A}}(h)$.

$$\begin{array}{c}
\frac{\mathcal{V} \vdash t_1 \equiv t_2}{\mathcal{V} \vdash t_2 \equiv t_1} \text{SYM} \\
\\
\frac{t_1 \in \text{Tm}_\Sigma(\mathcal{V})}{\mathcal{V} \vdash t_1 \equiv t_1} \text{REFL} \\
\\
\frac{\mathcal{V} \vdash t_1 \equiv t_2 \quad \mathcal{V} \vdash t_2 \equiv t_3}{\mathcal{V} \vdash t_1 \equiv t_3} \text{TRANS} \\
\\
\frac{\mathcal{V} \vdash t_1 \equiv t_2 \quad \mathcal{V}' \cup \{x\} \vdash C_1 \equiv C_2}{\mathcal{V} \cup \mathcal{V}' \vdash C_1[t_1/x] \equiv C_2[t_2/x]} \text{CTXT}
\end{array}$$

Figure 2.3: Congruence rules for $\equiv_{\mathbb{E}}$

Equational theories. Following [FH07], a (syntactic) *equation* over Σ is

$$\mathcal{V} \vdash t_1 \equiv t_2$$

where \mathcal{V} is a finite set of variables and $t_1, t_2 \in \text{Tm}_\Sigma(\mathcal{V})$. An *equational theory* then is $\mathbb{T} = (\Sigma, \mathbb{E})$ where \mathbb{E} is a set of equations on Σ .

An algebra $(X, \llbracket - \rrbracket_X)$ for Σ *satisfies an equality* $\mathcal{V} \vdash t_1 \equiv t_2$ if for every instance $\rho \in X^\mathcal{V}$, $\llbracket t_1 \rrbracket_X(\rho) = \llbracket t_2 \rrbracket_X(\rho)$. We say that $(X, \llbracket - \rrbracket_X)$ is an *algebra for the equational theory* $\mathbb{T} = (\Sigma, \mathbb{E})$ (say \mathbb{T} -algebra) if it is an algebra for Σ that satisfies all equalities in \mathbb{E} . Similarly, a $(\mathcal{V}, \mathbb{T})$ -*algebra* is the data of a \mathbb{T} -algebra $\mathcal{A} \in \mathbb{T}\text{-Alg}$ together with a function $h : \mathcal{V} \rightarrow \mathcal{A}$.

Free \mathbb{T} -algebras over a set \mathcal{V} also exist. For $t_1, t_2 \in \text{Tm}_\Sigma(\mathcal{V})$, define $t_1 \equiv_{\mathbb{E}} t_2$ iff it is provable using all equalities in \mathbb{E} and the congruence rules depicted on figure 2.3. This generates an equivalence relation on $\text{Tm}_\Sigma(\mathcal{V})$ whose equivalence classes are written $\{t\}_{\mathbb{E}} \in \text{Tm}_\Sigma(\mathcal{V})/\equiv_{\mathbb{E}}$ for every $t \in \text{Tm}_\Sigma(\mathcal{V})$. By definition, the set $\text{Tm}_\Sigma(\mathcal{V})/\equiv_{\mathbb{E}}$ with interpretation functions $\llbracket s \rrbracket(x_1 \dots x_n) = \{(s) x_1 \dots x_n\}_{\mathbb{E}}$ is a \mathbb{T} -algebra, denoted $\mathbb{T}(\mathcal{V})$. Besides, there is a direct embedding $\text{id} : \mathcal{V} \rightarrow \mathbb{T}(\mathcal{V})$ and for every other $(\mathcal{V}, \mathbb{T})$ -algebra $h : \mathcal{V} \rightarrow \mathcal{U}(A)$, the function

$$\begin{array}{ccc}
h^* : \mathbb{T}(\mathcal{V}) & \rightarrow & \mathcal{U}(A) \\
\bar{t} & \mapsto & \llbracket t \rrbracket_A
\end{array}$$

is a well defined morphism of \mathbb{T} -algebras and is the only one that makes the diagram below commute:

$$\begin{array}{ccc}
\text{Tm}_\Sigma(\mathcal{V})/\equiv_{\mathbb{E}} & \xrightarrow{h^*} & \mathcal{U}(A) \\
\text{id} \swarrow & & \nearrow h \\
& \mathcal{V} &
\end{array}$$

As a result we have:

Lemma 2.5. *Given a set \mathcal{V} , $\mathbb{T}(\mathcal{V}) = (\text{Tm}_\Sigma(\mathcal{V})/\equiv_{\mathbb{E}}, \{-\}_{\mathbb{E}})$ is the free \mathbb{T} -algebra over \mathcal{V} .*

Inequational theories. Adapting definitions from those of equational theories, an *inequation* over a signature Σ is

$$\mathcal{V} \vdash t_1 \leq t_2$$

where \mathcal{V} is a set of variables; and an *inequational theory* is $\mathbb{T} = (\Sigma, \mathbb{E})$ where \mathbb{E} is a set of inequations on Σ .

To reflect inequations from \mathbb{T} into algebras, we now ask the carrier set of an algebra $(X, \llbracket - \rrbracket_X)$ to be equipped with a partial order \leq_X such that every interpretation function $\llbracket s \rrbracket_X : X^n \rightarrow X$ is non-decreasing with respect to \leq_X . Then an algebra *satisfies all inequalities* in \mathbb{E} if for every $\mathcal{V} \vdash t_1 \leq t_2$ and every instance $\rho \in X^\mathcal{V}$, $\llbracket t_1 \rrbracket_X(\rho) \leq_X \llbracket t_2 \rrbracket_X(\rho)$.

With that definition, the *free \mathbb{T} -algebra* over a set \mathcal{V} still exists; also denoted $\mathbb{T}(\mathcal{V})$, it is constructed in a similar fashion than for equational theories. First, the inequalities in \mathbb{E} together with the congruence rules on figure 2.3 where \leq replaces \equiv , and SYM has been deleted define a preorder \leq_E on $\text{Tm}_\Sigma(\mathcal{V})$. This induces an equivalence relation

$$t_1 \equiv_{\mathbb{E}} t_2 \quad \text{iff} \quad t_1 \leq_{\mathbb{E}} t_2 \text{ and } t_2 \leq_{\mathbb{E}} t_1$$

such that \leq_E is a partial order on the quotient set $\text{Tm}_\Sigma(X)/\equiv_{\mathbb{E}}$, still written $\leq_{\mathbb{E}}$.

Lemma 2.6. *Given a set \mathcal{V} , $\mathbb{T}(\mathcal{V}) = (\text{Tm}_\Sigma(X)/\equiv_{\mathbb{E}}, \llbracket - \rrbracket)$ with partial order $\leq_{\mathbb{E}}$ is the free \mathbb{T} -algebra over \mathcal{V} .*

Proof. This is a routine check similar to what is done for equational theories. Key points are that the CTXT rule allows for the interpretation function to be well defined and non-decreasing. The non-decreasing assumption for other algebras then implies freeness of $\mathbb{T}(\mathcal{V})$. \square

Given an equational theory $\mathbb{T} = (\Sigma, \mathbb{E})$, the inequational theory $\mathbb{T}' = (\Sigma, \mathbb{E}')$ with

$$\mathcal{V} \vdash t_1 \leq_{E'} t_2 \quad \text{iff} \quad \mathcal{V} \vdash t_1 \equiv_E t_2$$

has the same free algebra than \mathbb{T} . In that regards equational theories are subsumed by inequational theories and we will focus on free algebras for inequational theories in the coming subsection.

Note also that free \mathbb{T} -algebra *support substitution*: for every substitution $\gamma : \mathcal{V}_1 \rightarrow \mathcal{V}_2$, we have $\{t\}_{\mathbb{E}}[\gamma] = \{t[\gamma]\}$.

2.2.2 A category of \mathbb{T} -strategies

In their full generality, strategies can be annotated with elements of any free \mathbb{T} -algebra:

Definition 2.8. Let $\mathbb{T} = (\Sigma, \mathbb{E})$ be an inequational theory, a \mathbb{T} -strategy is the data of a plain strategy $\sigma : S \multimap A$ together with a \mathbb{T} -annotation

$$\lambda_\sigma : (s \in S^+) \rightarrow \mathbb{T}([s]^-)$$

We write $\sigma : A \xrightarrow{\mathbb{T}} B$ for \mathbb{T} -strategies from A to B .

Similarly to plain and Σ -strategies, \mathbb{T} -strategies will be considered up to isomorphism. As in definition 2.4, two \mathbb{T} -strategies are *isomorphic* if they are isomorphic as plain strategies and their annotations are equal (after appropriate variable renaming). More generally \mathbb{T} -strategies also admit a partial ordering coherent with equality up to isomorphism:

Definition 2.9. Let σ, σ' be two \mathbb{T} -strategies on the same game, then $\sigma \leq \sigma'$ iff $\sigma \stackrel{\varphi}{\simeq} \sigma'$ as plain strategies and $\varphi.\lambda_\sigma \leq_E \lambda_{\sigma'}$.

It is direct to check that this defines a partial order such that $\sigma \leq \sigma'$ and $\sigma' \leq \sigma$ implies $\sigma \simeq \sigma'$.

\mathbb{T} -strategies as equivalence classes of Σ -strategies. The intuition behind the definition of \mathbb{T} -annotations for strategies is very similar to the one described for Σ -augmentations. In fact Σ -strategies are just a particular case of \mathbb{T} -strategies where $\equiv_{\mathbb{E}}$ is reflexivity. But the connection runs deeper: for $\mathbb{T} = (\Sigma, \mathbb{E})$, one can view \mathbb{T} -strategies as equivalence classes of Σ -strategies under $\equiv_{\mathbb{E}}$. This permits \mathbb{T} -strategies to inherit from all the categorical structure of Σ -CG without redefining composition, tensoring or copycat strategies from scratch. The rest of this section makes the connection between \mathbb{T} -strategies and Σ -strategies explicit and details the kind of constructions we get from it.

The equivalence relation $\equiv_{\mathbb{E}}$ and partial order $\leq_{\mathbb{E}}$ extend to Σ -strategies:

Definition 2.10. Let $\mathbb{T} = (\Sigma, \mathbb{E})$ be an inequational theory and let σ, σ' be two Σ -strategies on the same game then $\sigma \equiv_{\mathbb{E}} \sigma'$ (respectively $\sigma \leq_{\mathbb{E}} \sigma'$) iff $\sigma \stackrel{\varphi}{\simeq} \sigma'$ as plain strategies and $\varphi.\lambda_\sigma \equiv_{\mathbb{E}} \lambda_{\sigma'}$ (respectively $\leq_{\mathbb{E}}$).

It is again immediate that this defines an equivalence relation and a partial order over the corresponding equivalence classes.

Proposition 2.4. For A a game, there are inverse order-isomorphisms, $\overline{(-)}$ and $\{-\}_{\mathbb{E}}$, between

- \mathbb{T} -strategies over A ordered by \leq , and,
- $\equiv_{\mathbb{E}}$ -equivalence classes of Σ -strategies over A ordered by $\leq_{\mathbb{E}}$

Proof. Given a Σ -strategy $\sigma : A$, it can be turned into a \mathbb{T} -strategy $\{\sigma\}_{\mathbb{E}}$ having the same plain structure as σ and annotation $\lambda_{\{\sigma\}_{\mathbb{E}}}(s^+) := \{\lambda_{\sigma}(s^+)\}_{\mathbb{E}}$. One can then check that this preserves $\cong_{\mathbb{E}}$ and $\leq_{\mathbb{E}}$.

Conversely, given a \mathbb{T} -strategy $\tau : A$, it can be turned into a Σ -strategy $\overline{\tau}$ by picking a representative for every $\lambda_{\tau}(s^+)$. This also preserves order and for every Σ -strategy $\sigma : A$ and \mathbb{T} -strategy $\tau : A$

$$\overline{\{\sigma\}_{\mathbb{E}}} \equiv_{\mathbb{E}} \sigma \quad \{\overline{\tau}\}_{\mathbb{E}} = \tau$$

□

Thanks to this correspondence, it is possible to derive a composition and a tensor product from Σ -strategies to \mathbb{T} -strategies.

Definition 2.11. For $\sigma : A \xrightarrow{\mathbb{T}} B$, $\tau : B \xrightarrow{\mathbb{T}} C$ and $\rho : C \xrightarrow{\mathbb{T}} D$ we set

$$\sigma \odot \tau := \{\overline{\sigma} \odot \overline{\tau}\}_{\mathbb{E}} \quad , \quad \alpha_A := \{\alpha_A\}_{\mathbb{E}} \quad \text{and} \quad \sigma \otimes \rho := \{\overline{\sigma} \otimes \overline{\rho}\}_{\mathbb{E}}$$

As a result, $\lambda_{\tau \odot \sigma} = (\lambda_{\tau \otimes \sigma}) \upharpoonright_{(T \odot S)^+}$ with $\lambda_{\tau \otimes \sigma}$ following the same equations as in lemma 2.3 and similarly for $\lambda_{\sigma \otimes \tau} = \lambda_{\sigma} \parallel \lambda_{\tau}$

These constructions are well-defined as $\equiv_{\mathbb{E}}$ and $\leq_{\mathbb{E}}$ are congruences with respect to \odot and \otimes :

Proposition 2.5. Let $\sigma, \sigma' : A \xrightarrow{\Sigma} B$ and $\tau, \tau' : B \xrightarrow{\Sigma} C$ such that $\sigma \leq_{\mathbb{E}} \sigma'$ and $\tau \leq_{\mathbb{E}} \tau'$, then

$$\sigma \odot \tau \leq_{\mathbb{E}} \sigma' \odot \tau'$$

And similarly with $\equiv_{\mathbb{E}}$ replacing $\leq_{\mathbb{E}}$.

Proof. We detail only the case where $\sigma \leq_{\mathbb{E}} \sigma'$ and $\tau = \tau'$. The case for $\sigma = \sigma'$ and $\tau \leq_{\mathbb{E}} \tau'$ is symmetric and the general case then follows by transitivity. Congruence for $\equiv_{\mathbb{E}}$ is a direct consequence.

By definition $\sigma \stackrel{\phi}{\simeq} \sigma'$ such that $\phi.\lambda_{\sigma} \leq_{\mathbb{E}} \lambda_{\sigma'}$, and, by proposition 1.2 there is $\tau \otimes \sigma \stackrel{T \otimes \phi}{\simeq} \tau \otimes \sigma'$ such that

$$\begin{array}{ccccc}
 & & T \otimes \phi & & \\
 & & \curvearrowright & & \\
 & T \otimes S & & & T \otimes S' \\
 \Pi_1 \swarrow & & \phi \parallel C & & \Pi_1 \swarrow \\
 S \parallel C & \xrightarrow{\Pi_2} & A \parallel T & \xrightarrow{\sigma' \parallel C} & S' \parallel C & \xrightarrow{\Pi_2} & A \parallel T \\
 & \searrow \sigma \parallel C & \searrow A \parallel \tau & & \searrow \sigma' \parallel C & \searrow A \parallel \tau & \\
 & & A \parallel B \parallel C & & & &
 \end{array}$$

By well-founded induction on $<_{\tau \otimes \sigma'}$ and the characterisation of annotations for interaction it is then easy to check that

$$(T \otimes \phi). \lambda_{\tau \otimes \sigma} \leq \lambda_{\tau \otimes \sigma'}$$

the key being that \leq_E is a congruence at the level of terms. We detail the case for $\llbracket s', t \rrbracket_{\varphi'} \in T \otimes S'$ and $s \in S^+$ – other cases are simpler. Let $\phi^{-1}(\llbracket s', t \rrbracket_{\varphi'}) = \llbracket s, t \rrbracket_{\varphi}$ then

$$\begin{aligned} & (T \otimes \phi). \lambda_{\tau \otimes \sigma}(\llbracket s, t \rrbracket_{\varphi}) \\ &= \lambda_{\tau \otimes \sigma}(\llbracket s, t \rrbracket_{\varphi}) \quad [T \otimes \phi] \\ &= \lambda_{\sigma}(s) [\Pi_{1, \varphi}^{-1}] [\lambda_{\tau \otimes \sigma}] \quad [T \otimes \phi] && \text{(Lemma 2.3)} \\ &= \lambda_{\sigma}(s) [\Pi_{1, \varphi}^{-1}] [T \otimes \phi] [T \otimes \phi. \lambda_{\tau \otimes \sigma}] \\ &= \lambda_{\sigma}(s) \quad [\phi] [\Pi_{1, \varphi'}^{-1}] \quad [T \otimes \phi. \lambda_{\tau \otimes \sigma}] && \text{(Above diagram)} \\ &\leq_{\mathbb{E}} \lambda_{\sigma'}(s') \quad [\Pi_{1, \varphi'}^{-1}] \quad [T \otimes \phi. \lambda_{\tau \otimes \sigma}] && \left(\begin{array}{l} \sigma \leq \sigma' \\ + \text{CTXT} \end{array} \right) \\ &\leq_{\mathbb{E}} \lambda_{\sigma'}(s') \quad [\Pi_{1, \varphi'}^{-1}] \quad [\lambda_{\tau \otimes \sigma'}] && \left(\begin{array}{l} \text{Induction} \\ + \text{CTXT} \end{array} \right) \\ &= \lambda_{\tau \otimes \sigma'}(\llbracket s', t \rrbracket_{\varphi'}) && \text{(Lemma 2.3)} \end{aligned}$$

In the above, the CTXT rule is used several times in a row in order to apply the corresponding substitutions. This is possible as every of these substitutions are idempotent so, a variable that has been substituted does not appear any longer in the term. \square

Note that although the non-decreasing condition on interpretation functions is not necessary to define a category of \mathbb{T} -algebras with free objects – taking the more restrictive condition $\mathcal{V} \vdash t_1 \equiv_{\mathbb{E}} t_2$ in the CTXT rule –, it is essential here for \odot to preserve \leq_E . The example below shows how \leq_E may fail to be a congruence, interpreting terms as decreasing functions over reals.

Example 2.2. Consider the following two strategies with decreasing annotations on \mathbb{R} :

$$\begin{array}{ccc} \sigma : A & \multimap B & \leq \quad \sigma' : A & \multimap B \\ \bullet^0 \circ & \searrow & \bullet^1 \circ & \searrow \\ & \triangle & & \triangle \\ & \diamond^{(-2) \cdot \circ} & & \diamond^{(-1) \cdot \circ} \end{array}$$

then, their composition with

$$\tau : A^\perp$$

$$\bullet$$

$$\downarrow$$

$$\circ$$

yields

$$\tau \odot \sigma : B \quad \not\leq \quad \tau \odot \sigma' : B$$

$$\diamond^0 \qquad \qquad \qquad \diamond^{-1}$$

Proposition 2.6. *Let $\sigma, \sigma' : A \xrightarrow{\Sigma} B$ and $\tau, \tau' : C \xrightarrow{\Sigma} D$ such that $\sigma \leq_{\mathbb{E}} \sigma'$ and $\tau \leq_{\mathbb{E}} \tau'$, then*

$$\sigma \otimes \tau \leq_{\mathbb{E}} \sigma' \otimes \tau'$$

And similarly with $\equiv_{\mathbb{E}}$ replacing $\leq_{\mathbb{E}}$.

Proof. This is fairly straightforward. We focus again on the case where $\sigma \stackrel{\phi}{\simeq} \sigma'$ and $\tau = \tau'$. Then by proposition 1.8, \simeq is a congruence with respect to \otimes on plain strategies and using the congruence rules of $\leq_{\mathbb{E}}$

$$(\phi \otimes T). \lambda_{\sigma \otimes \tau} \leq_{\mathbb{E}} \lambda_{\sigma' \otimes \tau}$$

For example $\lambda_{\sigma \otimes \tau}(0, s^+)[\phi \otimes T] = \lambda_{\sigma}(s^+)[\iota_0][\phi \otimes T] = \lambda_{\sigma}(s^+)[\phi][\iota_0] \leq_{\mathbb{E}} \lambda_{\sigma'}(\phi(s^+))[\iota_0] = \lambda_{\sigma' \otimes \tau'}(0, \phi(s^+)) = \lambda_{\sigma' \otimes \tau'}((\phi \otimes T)(0, s^+))$. \square

These two lemmas ensure that the above constructions are well-defined. Other structural morphisms are also obtained as the images of structural morphisms in Σ -CG via $\{-\}_{\mathbb{E}}$. By theorem 2.3 we have:

Theorem 2.4. *Games and \mathbb{T} -strategies up to isomorphism, together with \odot , $(1, \otimes)$ and $(_)^\perp$ form a compact closed category \mathbb{T} -CG that is order enriched over $\leq_{\mathbb{E}}$.*

Preserving $\leq_{\mathbb{E}}$ is desirable in some cases, in particular in part III of this thesis where we will compare strategies whose annotations correspond to the resource analysis of some concurrent programs.

2.3 Examples of \mathbb{T} -strategies

In this section we instantiate the construction $\mathbb{T}\text{-CG}$ with several examples of interest. We first build a category of games and strategies annotated by (functions over) reals that will be used in part III. Then we extend the definition of $\mathbb{T}\text{-CG}$ in order for inequational theories to have a many-sorted signature. This allows us to build a category of games and strategies annotated with morphisms from any cartesian category. We show that the original category embeds in the resulting category. This last case is first introduced through the particular case of the category Set .

2.3.1 $\text{CG}_{\mathbb{R}}$

This first instantiation of $\mathbb{T}\text{-CG}$ is a very simple one that results in a category of concurrent games in which strategies provide a real (or a real function) together with every positive events they play and such that their composition composes these annotations.

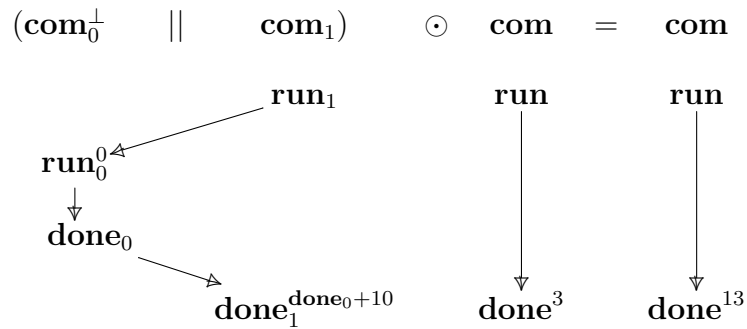
Definition 2.12. A \mathbb{R} -strategy over a game A is a plain strategy $\sigma : S \rightarrow A$ together with an annotation function

$$\lambda_{\sigma}(s) : \mathbb{R}^{[s]^+} \rightarrow \mathbb{R}$$

for any $s \in S$ with positive polarity.

Example 2.3. Let $\mathbf{com} : \mathbf{run}^- \rightarrow \mathbf{done}^+$ be a game representing the type of commands of a programming language, let $P : \mathbf{com}$ be some command that takes 3 seconds to execute and $\text{delay10} : \mathbf{com} \rightarrow \mathbf{com}$ be a function that delays the response of a command by 10 seconds. The interpretation of their sequential composition in term of \mathbb{R} -strategies will be of the form

$$\text{delay10} \odot P =$$



where positive events are annotated by the term-expression corresponding to their function, *e.g.* $\mathbf{done}_1 + 10$ stands for the function $\mathbf{run}_2, \mathbf{done}_1 \mapsto \mathbf{done}_1 + 10$.

Let us show that \mathbb{R} -strategies can be described in terms of \mathbb{T} -strategies, for a well chosen \mathbb{T} .

Let $\underline{\mathbb{R}}$ be the signature that contains a k -ary symbol for every k -ary real function:

$$\mathbf{f} \in \underline{\mathbb{R}}^k \quad \text{iff} \quad f : \mathbb{R}^k \rightarrow \mathbb{R}$$

\mathbb{R} is an obvious algebra for $\underline{\mathbb{R}}$ (with interpretation functions $\llbracket \mathbf{f} \rrbracket = f$) and, given a set of variables \mathcal{V} , freeness provides a canonical interpretation of $\text{Tm}_{\underline{\mathbb{R}}}(\mathcal{V})$ for every *valuation* $\rho : \mathbb{R}^{\mathcal{V}}$, written $\llbracket - \rrbracket_{\rho} : \text{Tm}_{\underline{\mathbb{R}}}(\mathcal{V}) \rightarrow \mathbb{R}$. Defining $\mathbb{E}_{\underline{\mathbb{R}}}$ to be the set of equations

$$\mathcal{V} \vdash t_1 = t_2 \quad \text{iff} \quad \forall \rho : \mathcal{V} \rightarrow \mathbb{R}, \llbracket t_1 \rrbracket_{\rho} = \llbracket t_2 \rrbracket_{\rho}$$

we have:

Proposition 2.7. *The set of \mathbb{R} -strategies over a game A is isomorphic to the set of $(\underline{\mathbb{R}}, \mathbb{E}_{\underline{\mathbb{R}}})$ -strategies over this same game.*

Proof. The quotient of $\text{Tm}_{\underline{\mathbb{R}}}$ by $\equiv_{\mathbb{E}_{\underline{\mathbb{R}}}}$ induces the following two mappings to be inverse from each other:

- For $\sigma : S \rightarrow A$ a \mathbb{R} -strategy define $\underline{\sigma} : S \rightarrow A$ the $(\underline{\mathbb{R}}, \mathbb{E}_{\underline{\mathbb{R}}})$ -strategy with same plain structure as σ and annotations for every $s \in S^+$ the equivalence class of the symbolic representation of $\lambda_{\sigma}(s^+)$, that is $\lambda_{\underline{\sigma}}(s^+) = \{(\lambda_{\sigma}(s^+)) s_1 \dots s_n\}_{\mathbb{E}}$ for s_1, \dots, s_n the negative variables associated with $[s]_S^-$.
- Conversely, let $\tau : T \rightarrow A$ be a $(\underline{\mathbb{R}}, \mathbb{E}_{\underline{\mathbb{R}}})$ -strategy, then one can define $\bar{\tau} : T \rightarrow A$ the \mathbb{R} -strategy with same plain structure and interpreted labels $\lambda_{\bar{\tau}}(t^+) (\rho : \mathbb{R}^{[t^+]^-}) = \llbracket \lambda_{\tau}(t^+) \rrbracket_{\rho}$.

□

Relying on proposition 2.7, one can transport the categorical structure of $(\underline{\mathbb{R}}, \mathbb{E}_{\underline{\mathbb{R}}})$ -CG onto \mathbb{R} -strategies:

Definition 2.13. For $\sigma : A \dashrightarrow B$, $\tau : B \dashrightarrow C$ and $\rho : C \dashrightarrow D$ three \mathbb{R} -strategies we set

$$\sigma \circ \tau := \overline{(\underline{\sigma} \circ \underline{\tau})} \quad , \quad \mathfrak{c}_A := \overline{(\underline{\mathfrak{c}}_A)} \quad \text{and} \quad \sigma \otimes \rho := \overline{(\underline{\sigma} \otimes \underline{\rho})}$$

Corollary 2.2. *Games and \mathbb{R} -strategies from a compact closed category called \mathbb{R} -CG.*

From definition 2.5, 2.11 and 2.13, one can note in particular that \mathbb{R} -annotations for an interaction $\tau \otimes \sigma : A$ are described inductively on the causal history of pairs of synchronised events by

$$\begin{aligned}\lambda_{\tau \otimes \sigma}(\llbracket s^+, t^- \rrbracket_\varphi) &= \lambda_\sigma(s) \circ \langle \lambda_{\tau \otimes \sigma}(e) \rangle_{e \in \llbracket s, t \rrbracket_{\varphi, S}^-} \\ \lambda_{\tau \otimes \sigma}(\llbracket s^-, t^+ \rrbracket_\varphi) &= \lambda_\tau(t) \circ \langle \lambda_{\tau \otimes \sigma}(e) \rangle_{e \in \llbracket s, t \rrbracket_{\varphi, T}^-}\end{aligned}$$

where $\llbracket s, t \rrbracket_{\varphi, S}^- = \{e \in \llbracket s, t \rrbracket_\varphi \mid \pi_1(e) \in [s]^-\}$, $\llbracket s, t \rrbracket_{\varphi, T}^- = \{e \in \llbracket s, t \rrbracket_\varphi \mid \pi_2(e) \in [t]^-\}$ and where projections and variable renaming for matching domains are kept silent.

Following the same reasoning, one can construct \mathbb{R}^{\geq} -CG, the category of games and strategies annotated with non-decreasing real functions. This category is equipped with a partial order

$$\sigma \leq \tau : A \quad \text{iff} \quad \sigma \stackrel{\mathcal{L}}{\simeq} \tau \quad \text{and} \quad \forall s \in S^+, \lambda_\sigma(s) \leq \lambda_\tau(\varphi(s))$$

This is achieved by showing equivalence with strategies over the inequational theory $(\underline{\mathbb{R}}^{\geq}, \mathbb{E}_{\mathbb{R}^{\geq}})$ defined by

$$\mathbf{f} \in (\underline{\mathbb{R}}^{\geq})^k \quad \text{iff} \quad f : \mathbb{R}^k \rightarrow \mathbb{R} \quad \text{is non-decreasing}$$

and

$$\mathcal{V} \vdash t_1 \leq t_2 \quad \text{iff} \quad \forall \rho : \mathcal{V} \rightarrow \mathbb{R}, \llbracket t_1 \rrbracket_\rho \leq \llbracket t_2 \rrbracket_\rho$$

2.3.2 CG_{Set}

In the previous example, annotations on strategies are all real functions. Yet, it might be desirable to have annotations with non uniform domains and codomains, that is have general set morphisms as annotations. For the interaction to be possible in this context, Player and Opponent must agree on the domains and codomain of their annotations. For that, we enrich games with *typing*, a new rule that specifies for every event a set in which its annotations must lay and that must be preserved by players.

Definition 2.14. Let A be a plain game, a *Set-typing* for A is a mapping $\theta : A \rightarrow \text{Set}$. We note $\theta_a = \theta(a)$ for the *type* of $a \in A$. This extends to any subset $X \subseteq A$ by taking $\theta_X = \prod_{a \in X} \theta_a$.

Annotations for positive events in a strategy then correspond to functions that respect this typing:

Definition 2.15. A *Set-strategy* over a Set-typed game (A, θ) is a plain strategy $\sigma : S \rightarrow A$ together with an annotation function

$$\lambda_\sigma : (s \in S^+) \rightarrow \text{Set}(\theta_{[s]^-}, \theta_s)$$

where θ_s is a shortcut for $\theta_{\sigma(s)}$ for every $s \in S$.

As with \mathbb{R} -CG, we aim at defining a category of such games and strategies, where composition performs composition of the corresponding annotations. To replay the trick of regarding Set as an equational theory $\underline{\text{Set}}$, we first allow signatures to have sorts.

Definition 2.16. A *many-sorted* signature (\mathcal{S}, Σ) is a class of sorts \mathcal{S} together with a set of symbol declarations $\mathfrak{s} \in \Sigma(\prod_{i=1}^n \theta_i, \theta)$ where $(\prod_{i=1}^n \theta_i, \theta)$ is the *type scheme* of \mathfrak{s} where $n \geq 0$ and $\theta_i \in \mathcal{S}$.

The use of classes instead of sets for sorts in the above definition allows us to consider large signature that would not be well-defined otherwise. A typical example is the signature $\underline{\text{Set}}$ of sorts $\mathcal{S} = \text{Set}_0$ and symbols:

$$\mathfrak{f} \in \underline{\text{Set}}(\prod_{i=1}^n \theta_i, \theta) \quad \text{iff} \quad f : \prod_{i=1}^n \theta_i \rightarrow \theta$$

Given a set \mathcal{V} of *sorted variables* – *i.e.* an indexed family of variables $\mathcal{V} = \{\mathcal{V}_\theta\}_{\theta \in \mathcal{S}}$ – the set of *sorted terms* over \mathcal{V} is the indexed family of terms inductively defined by

$$\begin{aligned} \text{Tm}_\Sigma^\theta(\mathcal{V}) &:= \mathcal{V}_\theta \cup \\ &\quad \left\{ (\mathfrak{s}) t_1 \dots t_k \mid \mathfrak{s} \in \Sigma(\prod_{i=1}^k \theta_i, \theta) \text{ and } t_i \in \text{Tm}_\Sigma^{\theta_i}(\mathcal{V}) \right\} \end{aligned}$$

Similarly, algebras can be extended to many-sorted signature by requiring an *indexed family* of carrier sets $X = \{X_\theta\}_{\theta \in \mathcal{S}}$ together with interpretation functions $\llbracket \mathfrak{s} \rrbracket : \prod_{i=1}^n X_{\theta_i} \rightarrow X_\theta$.

Sections 2.1 and 2.2 can be entirely rephrased in terms of typed terms and signatures. We can thus replay the equivalence between \mathbb{R} -strategies and $\underline{\mathbb{R}}$ -strategies from section 2.3.1, having instead Set-strategies and $\underline{\text{Set}}$ -strategies, with $\underline{\text{Set}}$ the many-sorted equational theory of signature $\underline{\text{Set}}$ and equations:

$$\mathcal{V} \vdash t_1 = t_2 \quad \text{iff} \quad \forall \rho = \{\rho_\theta : \mathcal{V}_\theta \rightarrow \theta\}_{\theta \in \mathcal{S}}, \llbracket t_1 \rrbracket_\rho = \llbracket t_2 \rrbracket_\rho$$

This induces a compact-closed category Set-CG. Moreover the remarks on \mathbb{R} -CG also hold for Set-CG, in particular it is possible to define strategies annotated with non-decreasing functions on partially ordered sets such that this induces a partial order enrichment for Set-CG.

2.3.3 \mathcal{C} -CG

Concluding our exploration of the expressiveness of \mathbb{T} -CG, we present our last construction, \mathcal{C} -CG. This is a natural generalisation of the previous two examples where annotations on strategies are morphisms from a cartesian category \mathcal{C} . In this model, as in Set -CG, every event of a game is associated with an object in \mathcal{C} representing the targeted “type of information” associated to this event. Strategies then provide for every Player move a morphism in \mathcal{C} that describes how this information relates to the one of its negative predecessors. We furthermore show that the category \mathcal{C} embeds in the resulting \mathcal{C} -CG.

Let $(\mathcal{C}, \times, \mathbf{1})$ be a cartesian category, we define \mathcal{C} -games and \mathcal{C} -strategies to be:

Definition 2.17. A \mathcal{C} -game is the data of a plain game A together with a \mathcal{C} -typing for A , that is, a mapping

$$\theta : A \rightarrow \mathcal{C}_0$$

where \mathcal{C}_0 is the class of objects of \mathcal{C} . We note $\theta_a = \theta(a)$ for the *type* of $a \in A$. This extends to any subset $X \subseteq A$ by taking $\theta_X = \prod_{a \in X} \theta_a$.

Definition 2.18. A \mathcal{C} -strategy over a \mathcal{C} -game (A, θ) is the data of a plain strategy $\sigma : S \rightarrow A$ together with a \mathcal{C} -annotation for σ that respects θ , that is, a mapping

$$\lambda_\sigma : (s \in S^+) \rightarrow \mathcal{C}_1([s]^-, s)$$

where $\mathcal{C}_1([s]^-, s)$ is a shortcut for the set of morphisms $\mathcal{C}_1(\theta_{\sigma([s]^-)}, \theta_{\sigma(s)})$.

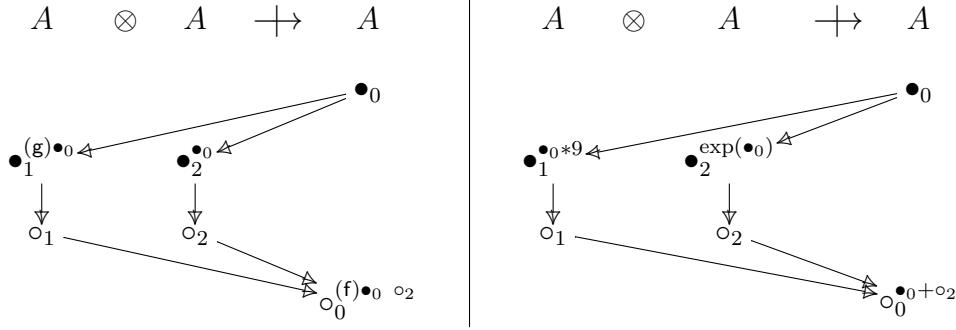
Example 2.4. Figures 2.4 show the same plain strategy with two different kinds of annotations: on the left, $\mathcal{C} = \text{Subst}_\Sigma$ (for $\Sigma \supseteq \{f^2, g^1\}$) and every event in A has type $\{*\}$, σ is a Subst_Σ -strategy but is in fact equivalent to a Σ -strategy; on the right, A is typed in Set with $\theta(\bullet) = \mathbb{R}$ and $\theta(\circ) = \mathbb{N}$ and σ is annotated with set-functions. In order to not overload the diagrams, both terms and functions are written as expressions.

Similarly to what we have done for \mathbb{R} and Set , we set $\underline{\mathcal{C}}$ to be the many-sorted equational theory with sorts $\mathcal{S} = \mathcal{C}_0$ and symbols

$$f \in \underline{\mathcal{C}}\left(\prod_{i=1}^n \theta_i, \theta\right) \quad \text{iff} \quad f \in \mathcal{C}_1\left(\prod_{i=1}^n \theta_i, \theta\right)$$

Then, one can inductively define a family of interpretation functions

$$\llbracket _ \rrbracket_{\mathcal{V}}^\theta : \text{Tm}_{\underline{\mathcal{C}}}^\theta(\mathcal{V}) \rightarrow \mathcal{C}\left(\prod_{v \in \mathcal{V}} \theta_v, \theta\right)$$

Figure 2.4: A Σ -strategy and a Set-strategy

such that $\llbracket v \rrbracket_{\mathcal{V}}^{\theta}$ is the projection map $\pi_v : \prod_{v' \in \mathcal{V}} \theta_{v'} \rightarrow \theta_v$ and

$$\llbracket (f) t_1 \dots t_k \rrbracket_{\mathcal{V}}^{\theta} = f \circ \langle \llbracket t_i \rrbracket_{\mathcal{V}}^{\theta_i} \rangle_{i=1}^k$$

with $\langle - \rangle$ the tupling operator associated with the cartesian structure of \mathcal{C} .

Then, taking equations $\mathbb{E}_{\mathcal{C}}$ to be

$$\mathcal{V} \vdash t_1 = t_2 : \theta \quad \text{iff} \quad \llbracket t_1 \rrbracket_{\mathcal{V}}^{\theta} = \llbracket t_2 \rrbracket_{\mathcal{V}}^{\theta}$$

the category $(\mathcal{C}, \mathbb{E}_{\mathcal{C}})$ -CG can be used to define composition, identities, and tensor product for \mathcal{C} -strategies.

Definition 2.19. For $\sigma : A \multimap B$, $\tau : B \multimap C$ and $\rho : C \multimap D$ three \mathcal{C} -strategies we set

$$\sigma \circ \tau := \llbracket \underline{\sigma} \circ \underline{\tau} \rrbracket \quad , \quad \mathbb{C}_A := \llbracket \underline{\mathbb{C}}_A \rrbracket \quad \text{and} \quad \sigma \otimes \rho := \llbracket \underline{\sigma} \otimes \underline{\rho} \rrbracket$$

where $\llbracket - \rrbracket$ and $\underline{(-)}$ are the set isomorphisms between \mathcal{C} -strategies and $(\mathcal{C}, \mathbb{E}_{\mathcal{C}})$ -strategies define as in proposition 2.7.

Corollary 2.3. *Games and \mathcal{C} -strategies form a compact closed category called \mathcal{C} -CG.*

From definition 2.5, 2.11 and 2.19, one can again note that for two \mathcal{C} -strategies $\sigma : A \xrightarrow{\mathcal{C}} B$, $\tau : B \xrightarrow{\mathcal{C}} C$, the \mathcal{C} -annotations for the interaction $\tau \otimes \sigma$ are described inductively on the causal history of pairs of synchronised events by $\lambda_{\tau \otimes \sigma}(\llbracket s, t \rrbracket_{\varphi}) =$

1. $\lambda_{\sigma}(s) \circ \langle \lambda_{\tau \otimes \sigma}(e) \rangle_{e \in (s, t)_{\varphi, S}^-}$ if $s \in S^+$;

2. $\lambda_\sigma(t) \circ \langle \lambda_{\tau \otimes \sigma}(e) \rangle_{e \in \{s,t\}_{\varphi, T}^-}$ if $t \in T^+$;
3. id_{θ_a} otherwise (for $\tau \otimes \sigma(\llbracket s, t \rrbracket_{\varphi}) = a$).

Interaction and composition of \mathcal{C} -strategies thus perform composition of their annotations as expected.

Reminiscent of Melliès' construction of the free dialogue category over a category [Mel12], \mathcal{C} -CG can be viewed as a compact closed completion of \mathcal{C} as the latest embeds as a sub-category in \mathcal{C} -CG:

Proposition 2.8. *There is a functor $[-] : \mathcal{C} \rightarrow \mathcal{C}\text{-CG}$ such that:*

- on object: $[-]$ maps $A \in \mathcal{C}_0$ to the positive singleton \mathcal{C} -game $[A] = \circ_A^+$
- on morphisms, $[-]$ maps $f : A \rightarrow B$ to the strategy

$$[-] : A \multimap B$$

$$\begin{array}{ccc} & \circ_A & \\ & \searrow & \\ & & \circ_B^{f(\circ_A)} \end{array}$$

Proof. By definition, $[\text{id}_A] = \mathfrak{c}_{[A]}$, and, from the above remark on annotations for composition it is immediate to check that $[f \circ g] \simeq [f] \odot [g]$. \square

The functor $[-]$ actually has all the data to define a lax monoidal functor but we have not checked yet whether these satisfy the required coherence diagram. These data are:

- the unique empty strategy $\emptyset : \mathbf{1} \multimap [\mathbf{1}]$;
- and for every $A, B \in \mathcal{C}_0$ the natural \mathcal{C} -strategies inverse from each other:

$$[A] \otimes [B] \multimap [A \times B]$$

$$\begin{array}{ccc} \circ_A & \circ_B & \\ & \searrow & \\ & & \circ_{A \times B}^{(\circ_A, \circ_B)} \end{array}$$

Together with the coherence diagram, it would be interesting to investigate if the completion $[-]$ could yield a free structure using some restriction on \mathcal{C} -strategies. These two questions are left open for future work.

Categorical structure of \mathbb{T} -CG

This chapter provides details on the proof of the compact closure of Σ -CG, the model of concurrent games and strategies with annotations as terms from a first order signature Σ . Following definition 2.11, this also provides a proof for the more general concurrent games model \mathbb{T} -CG enriched over an inequational theory \mathbb{T} . The proof for the compact closure of Σ -CG follows closely the structure devised in [CCRW17] for plain concurrent strategies.

First, we start by introducing a category of Σ -event structures and total morphisms between them. Similarly to Σ -strategies, Σ -event structures are equipped with term-annotations. However, these annotations are more general: they are defined on *configurations* rather than on events. This generalisation allows for the resulting category to have pullbacks. A crucial feature in the study of Σ -strategies and their interaction.

Second, we extend the framework of Σ -event structures with *partial* morphisms. This allows us to represent *hiding*, making explicit the connection between the interaction of Σ -strategies and their composition. Finally, this categorical view is used to show that Σ -CG is compact closed. In particular, this view helps in showing the *associativity* of composition and the *functoriality* of tensor.

As a conclusion, we discuss the possibility of exploiting the more general annotations on configurations of Σ -event structures in order to build an other model of games and annotated strategies in which annotations may actually affect the plain structure of their interaction. This model however is not fully developed as this generality was not necessary for the application cases considered in parts II and III.

3.1 A category of Σ -event structures

Chapter 2 gives a synthetic presentation of annotated strategies where annotations are provided with every positive events played in a strategy. Although this definition is handy to work with, it might seem a bit ad-hoc from an outsider's perspective. In this section we introduce a more primitive mathematical structure, Σ -event structure, that are to annotated strategies,

what plain event structure are to plain strategies. In particular they form a category with pullbacks. Hiding is however subject to restriction.

3.1.1 Σ -event structures

Although more primitive, Σ -event structures are defined following the same kind of ideas as Σ -strategies: every configuration carries additional information (terms) about its events and this information can be made more specific with further extensions.

Definition 3.1. Let Σ be a first order signature, a Σ -event structure is a plain event structure E together with an *annotation* λ_E that is an indexed family of *idempotent* substitutions

$$\{\lambda_E^x : x \xrightarrow{\Sigma} x\}_{x \in \mathcal{C}(E)}$$

such that (knowledge preservation) for every $x, x' \in \mathcal{C}(E)$

$$x \subseteq x' \implies \lambda_E^x \preceq (\lambda_E^{x'})|_x$$

Idempotence of annotation is for sanity: if the variable $e \in x$ belongs to $\text{fv}(\lambda^x(e))$ for some $e' \in x$ then e represents the *unknown* information associated with e , having $\lambda^x(e) \neq e$ would then not make sense. Similarly, knowledge preservation ensures that the piece of information attached to an event is *coherent* throughout extensions: only unknown or new variables can be made precise.

Lemma 3.1. For (E, λ_E) a Σ -event structure, knowledge preservation is equivalent to

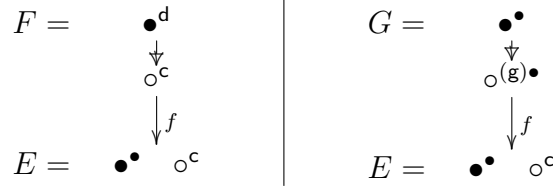
$$x \subseteq x' \in \mathcal{C}(E) \implies \lambda_E^{x'}|_x = \lambda_E^x[\lambda_E^{x'}|_x]$$

Proof. Right to left implication is immediate; left to right implication is a consequence of lemma 2.1 on idempotent substitutions. In particular, if ι denotes the inclusion $x \hookrightarrow x'$ then

$$\begin{array}{ccccc} x' & \xrightarrow{\lambda_E^{x'}} & x' & \xrightarrow{\iota} & x \\ & & \searrow \iota & & \nearrow \lambda_E^x \\ & & & x & \end{array}$$

□

Morphisms of event structures extend to Σ -annotations. In particular, for $f : E \rightarrow E$ a morphism of event structures, we make use of the fact that for every configuration $x \in \mathcal{C}(E)$, f defines an isomorphism $f : x \simeq f(x)$ (local injectivity) and can thus be used to transport (and compare) annotations:


 Figure 3.1: An example and a non-example of morphism of Σ -event structures

Definition 3.2. Let $(E, \lambda_E), (F, \lambda_F)$ be two Σ -event structures, $f : E \rightarrow F$ is a *morphism of Σ -event structures* (Σ -*morphism* for short) if it is a morphism of event structures and for every $x \in \mathcal{C}(E)$ (knowledge preservation)

$$\lambda_F^{f(x)} \preceq f.\lambda_E^x$$

Figure 3.1 shows two examples of the same morphism of event structures, extended with different annotations for the event structures involved. These annotations define valid Σ -event structures, however only the left example defines a valid morphism of Σ -event structures. In particular, the right example fails the above definition on the configuration $x = \{\bullet, \circ\}$: $\lambda_G^x(\circ) = (\mathbf{g})\bullet$ cannot be obtained as an instance of $\lambda_E^x(\circ) = \mathbf{c}$.

Generally speaking, a morphism of plain event structures $f : E \rightarrow F$ can be think as “ E is a specialisation of F ”; the last condition on Σ -morphisms follows this idea: through renaming by f , terms on E ’s configurations are instances of their F counterparts. In particular

Lemma 3.2. For $(E, \lambda_E), (F, \lambda_F)$ two Σ -event structures and $f : E \rightarrow F$ a plain morphism between them, then f is Σ -morphism iff

$$\lambda_F^{f(x)}[f.\lambda_E^x] = f.\lambda_E^x$$

Proof. Direct consequence of lemma 2.1 □

Parallel composition of event structures extends straightforwardly to Σ -event structures:

Definition 3.3. Let $(E, \lambda_E), (F, \lambda_F)$ be two Σ -event structures, $E \parallel F$ is the plain $E \parallel F$ event structure together with

$$\lambda_{E \parallel F}^{x_E \parallel x_F} := \lambda_E^{x_E} \parallel \lambda_F^{x_F}$$

Like in \mathcal{E} , the empty event structure is the unit for parallel composition, event structure morphisms compose and identities are neutral for composition, hence

Proposition 3.1. *Σ -event structures and their morphisms form a monoidal category $(\mathcal{E}_\Sigma, \parallel, \emptyset)$.*

Categories \mathcal{E} and \mathcal{E}_Σ are related to each other via two functors

$$\begin{array}{ccc} & \mathcal{F} & \\ \mathcal{E} & \xrightarrow{\quad} & \mathcal{E}_\Sigma \\ & \mathcal{G} & \end{array}$$

where $\mathcal{F} : \mathcal{E} \rightarrow \mathcal{E}_\Sigma$ is the functor that maps an event structure $E \in \mathcal{E}$ to the same event structure $E \in \mathcal{E}_\Sigma$ with annotation $\lambda(x \in \mathcal{C}(E)) = \text{id}_x$ – morphisms follow $-$, and $\mathcal{G} : \mathcal{E}_\Sigma \rightarrow \mathcal{E}$ is the forgetful functor from \mathcal{E}_Σ to \mathcal{E} . These two functors preserve the monoidal structure of both categories, however they do not define an adjunction: in the next section we show that \mathcal{G} does not preserve pullbacks.

3.1.2 Pullbacks

Similarly to the construction of plain concurrent strategies presented in chapter 1, Σ -strategies can be viewed as certain Σ -morphisms, this is useful in order to interpret interaction of strategies as pullbacks and thus inherit all good properties of pullbacks such as associativity. In this section we show that \mathcal{E}_Σ has pullbacks.

Via \mathcal{G} , morphisms of Σ -event structures define morphisms of event structures, so the plain structure of the pullback of two Σ -morphisms embeds in the pullback of the corresponding plain morphisms. In particular, for $\sigma : S \rightarrow A$ and $\tau : T \rightarrow A$, a configuration in $\sigma \wedge \tau$ must be a secured bijection. But, given such bijection $\varphi : x_S \simeq x_T$ what should then be its annotation? For Π_1 and Π_2 to define Σ -morphisms, this has to be an instance of both $\pi_1^{-1} \cdot \lambda_S^{x_S}$ and $\pi_2^{-1} \cdot \lambda_T^{x_T}$. In particular, because of idempotence, it must be a solution to the unification problem

$$\mathcal{U}_\varphi := \left\{ \lambda_S^{x_S}(s)[\pi_1^{-1}] \doteq \lambda_T^{x_T}(t)[\pi_2^{-1}] \right\}_{(s,t) \in \varphi}$$

Configurations in $S \wedge T$ must thus correspond to Σ -secured bijections, that are, secured bijections whose corresponding unification problem have a solution. We write $\mathcal{B}_{\sigma,\tau}^{\Sigma\text{-sec}}$ for the set of Σ -secured bijections. Because of knowledge preservation we have

Lemma 3.3. *If $\varphi \subseteq \varphi'$ are two secured bijections and φ' is Σ -secured, then φ is also Σ -secured.*

Proof. Let $\varphi : x_S \simeq x_T$, $\varphi' : x'_S \simeq x'_T$ and let $\mu : \varphi' \xrightarrow{\Sigma} \varphi$ be a unifier for $\mathcal{U}_{\varphi'}$, then, using lemma 3.1, it is simple to see that $((\pi_1^{-1} \cdot \lambda_S^{x'_S}) \circ \mu)_{\upharpoonright \varphi}$ ($= ((\pi_2^{-1} \cdot \lambda_T^{x'_T}) \circ \mu)_{\upharpoonright \varphi}$) is a unifier for \mathcal{U}_{φ} . Indeed, for every $(s, t) \in \varphi$:

$$\begin{aligned} \lambda_S^{x_S}(s)[\pi_1^{-1}][(\pi_1^{-1} \cdot \lambda_S^{x'_S} \circ \mu)_{\upharpoonright \varphi}] &= \lambda_S^{x'_S}(s)[\lambda_{S \upharpoonright x_S}^{x'_S}][\pi_1^{-1}][\mu_{\upharpoonright \varphi}] \\ &= \lambda_S^{x'_S}(s)[\pi_1^{-1}][\mu_{\upharpoonright \varphi}] \\ &= \lambda_T^{x'_T}(t)[\pi_2^{-1}][\mu_{\upharpoonright \varphi}] \\ &= \lambda_T^{x_T}(t)[\lambda_{T \upharpoonright x_T}^{x'_T}][\pi_2^{-1}][\mu_{\upharpoonright \varphi}] \\ &= \lambda_T^{x_T}(t)[\pi_2^{-1}][(\pi_2^{-1} \cdot \lambda_T^{x'_T} \circ \mu)_{\upharpoonright \varphi}] \end{aligned}$$

□

Lemma 3.3 entails that $\mathcal{B}_{\sigma, \tau}^{\Sigma\text{-sec}}$ is closed under inclusion. Inspired by the pullback construction for plain event structures (cf section 1.2.2), we now define the causal interaction of two Σ -morphisms:

Definition 3.4. Let $\sigma : S \rightarrow A$, $\tau : T \rightarrow A$ be two maps of Σ -events structures, their *causal interaction* $S \wedge_{\Sigma} T$ is the event structure defined by:

- Events: the prime Σ -secured bijections of $\mathcal{B}_{\sigma, \tau}^{\Sigma\text{-sec}}$
- Causality: inclusion of the graphs of the Σ -secured bijections,
- Consistency: X is a consistent set of prime Σ -secured bijections if the union of their graphs describes a Σ -secured bijection (*i.e.* $(\bigcup_{\varphi \in X} \varphi) \in \mathcal{B}_{\sigma, \tau}^{\Sigma\text{-sec}}$).

Checking that $S \wedge_{\Sigma} T$ defines a proper event structure is routine, especially using that $\mathcal{B}_{\sigma, \tau}^{\Sigma\text{-sec}}$ is down-closed. Moreover, there is an obvious inclusion of $S \wedge_{\Sigma} T$ into $S \wedge T$ at the level of plain event structures. Hence the projections $\Pi_1 : S \wedge_{\Sigma} T \rightarrow S$ and $\Pi_2 : S \wedge_{\Sigma} T \rightarrow T$ define as for $S \wedge T$ make the diamond diagram commutes in \mathcal{E} .

$$\begin{array}{ccc} & S \wedge_{\Sigma} T & \\ \Pi_1 \swarrow & & \searrow \Pi_2 \\ S & & T \\ \sigma \searrow & & \swarrow \tau \\ & A & \end{array}$$

Given $\varphi : x_S \simeq x_T \in \mathcal{B}_{\sigma, \tau}^{\Sigma\text{-sec}}$, we denote $\lambda^{\varphi} : \varphi \xrightarrow{\Sigma} \varphi$ the idempotent maximum substitution for $\pi_1^{-1} \cdot \lambda_S^{x_S}$ and $\pi_2^{-1} \cdot \lambda_T^{x_T}$ described in corollary 2.1, we have:

Lemma 3.4. *Let $\varphi \subseteq \varphi' \in \mathcal{B}_{\sigma, \tau}^{\Sigma\text{-sec}}$, then $\lambda^{\varphi}[\lambda_{\upharpoonright \varphi}^{\varphi'}] = \lambda_{\upharpoonright \varphi}^{\varphi'}$.*

Proof. Let $\varphi : x_S \simeq x_T$ and $\varphi' : x'_S \simeq x'_T$ such that $x'_S \subseteq x_S$ and $x'_T \subseteq x_T$. Then, by definition, $\lambda_{|\varphi}^{\varphi'}$ subsumes $(\pi_1^{-1}.\lambda_S^{x_S})_{|x'_S}$ and $(\pi_2^{-1}.\lambda_T^{x_T})_{|x'_T}$ so it also subsumes $\pi_1^{-1}.\lambda_S^{x'_S}$ and $\pi_2^{-1}.\lambda_T^{x'_T}$. Hence, $\lambda^{\varphi'} \preceq \lambda_{|\varphi}^{\varphi'}$ by minimality of $\lambda^{\varphi'}$. Hence, by lemma 2.1, $\lambda^\varphi[\lambda_{|\varphi}^{\varphi'}] = \lambda_{|\varphi}^{\varphi'}$. \square

As in the plain case, there is a strong correspondence between configurations of $S \wedge_\Sigma T$ and Σ -secured bijections.

Lemma 3.5. *For any configuration $x \in \mathcal{C}(S \wedge_\Sigma T)$, x defines a Σ -secured bijection $\varphi_x = \cup x : \Pi_1 x \simeq \Pi_2 x$. Moreover, the assignment $x \mapsto \varphi_x$ defines an order-isomorphism $\mathcal{C}(S \wedge_\Sigma T) \cong \mathcal{B}_{\sigma, \tau}^{\Sigma\text{-sec}}$ (with both sets ordered by inclusion), and there is a family of order-isomorphisms:*

$$\nu_x : \begin{array}{ccc} x & \simeq & \varphi_x \\ [s, t]_x & \mapsto & (s, t) \end{array}$$

that is natural in x .

Proof. This follows the exact same reasoning as for the plain case. We refer the reader to lemma 1.5. \square

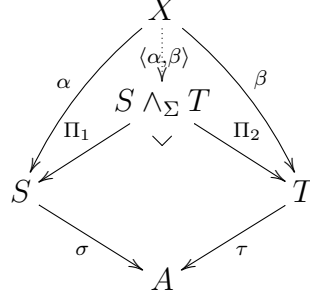
This lemma together with lemma 3.4 shows that the family of substitutions $\{\nu_x.\lambda^{\varphi_x}\}_{x \in \mathcal{C}(S \wedge T)}$ is an appropriate annotation for $S \wedge_\Sigma T$. In fact, this annotation turns the causal interaction $S \wedge_\Sigma T$ together with projections Π_1, Π_2 into a pullback in \mathcal{E}_Σ :

Proposition 3.2. *Let $\sigma : S \rightarrow A$, $\tau : T \rightarrow A$ be two Σ -morphisms, then the interaction $(S \wedge_\Sigma T, \Pi_1, \Pi_2)$ together with the family of annotations $\{\nu_x.\lambda^{\varphi_x}\}_{x \in \mathcal{C}(S \wedge T)}$ defines a pullback for σ and τ .*

Proof. First note that the diamond diagram pictured above still holds in \mathcal{E}_Σ . Let $x \in \mathcal{C}(S \wedge_\Sigma T)$, then by corollary 2.1 $\pi_1.\lambda^{\varphi_x} = \lambda_S^{x_S} \circ \pi_1.\lambda^{\varphi_x}$ and by definition $\Pi_1|_x = \pi_1 \circ \nu_x$. By lemma 3.2, this shows that Π_1 is knowledge preserving hence is a Σ -morphism. The case is similar for Π_2 .

Suppose now that (X, α, β) also satisfies the diamond diagram in \mathcal{E}_Σ . Then, we show that one can reuse the unique morphism of plain event structures $\langle \alpha, \beta \rangle : X \rightarrow S \wedge T$ obtained via the pullback property of $S \wedge T$ in \mathcal{E} to define a morphism of Σ -event structures that makes the following diagram

commutes:



Let $x \in \mathcal{C}(X)$ then $\langle \alpha, \beta \rangle(x) \in \mathcal{C}(S \wedge T)$ corresponds to the secured bijection $\varphi_x : \alpha(x) \simeq \beta(x)$, and, by lemma 3.2

$$\alpha_x^{-1} \cdot \lambda_S^{x_S} \circ \lambda_X^x = \lambda_X^x = \beta_x^{-1} \cdot \lambda_T^{x_T} \circ \lambda_X^x$$

Hence $\langle \alpha, \beta \rangle \cdot \lambda_X^x$ subsumes $(\pi_1^{-1} \cdot \lambda_S^{\alpha(x)})$ and $(\pi_2^{-1} \cdot \lambda_T^{\beta(x)})$ so, by minimality of λ^{φ_x} , we have $\lambda^{\varphi_x} \preceq \langle \alpha, \beta \rangle \cdot \lambda_X^x$. Uniqueness holds from uniqueness in \mathcal{E} . \square

From the existence of pullbacks, we define the *interaction* between any two Σ -morphisms $\sigma : A \parallel B$, $\tau : B \parallel C$ sharing a common component B by $\tau \otimes \sigma = (\sigma \parallel C) \wedge_{\Sigma} (A \parallel \tau)$. As in the plain case, configurations of $T \otimes S$ are fully described by pairs of configurations $x_T \otimes x_S$ such that $x_S \in \mathcal{C}(S)$, $x_T \in \mathcal{C}(T)$, $\sigma(x_S) = x_A \parallel x_B$, $\tau(x_T) = x_B \parallel x_C$ and $\varphi : x_S \parallel x_C \simeq x_A \parallel x_T$ is a Σ -secured bijection.

3.1.3 Partial morphisms and projections

Another crucial feature when building a framework for games and strategies is the ability to forget (or hide) part of the object under study. In \mathcal{E} this is simply done by means of projections. Projections however are not always defined in \mathcal{E}_{Σ} : let $E \in \mathcal{E}_{\Sigma}$ and $V \subseteq E$, then it may be that for $x \in \mathcal{C}(E)$, the restriction of λ_E^x to $x \cap V$ still has free variables in $E - V$. We say that V is a Σ -independent subset of E if for every $x \in \mathcal{C}(E)$, $\text{FV}(\lambda_{E \downarrow V}^x) \subseteq V$.

Definition 3.5. Let E be a Σ -event structure and V be a Σ -independent subset of E , the *projection* of E on V , written $E \downarrow V$, is defined by causal $E \downarrow V$ together with $\{\lambda_{E \downarrow V}^{[x]_E}\}_{x \in E \downarrow V}$.

This definition may seem a bit ad-hoc, it will be simplified in the next section when focusing on morphisms actually defining Σ -strategies. For the moment it is necessary to introduce the category of Σ -event structures and partial morphisms between them, and restate a useful lemma describing how projection and interaction commute, namely a zipping lemma, already known in the plain case [CCRW17].

The category $\mathcal{E}_{\Sigma, \perp}$ Our category of partial Σ -morphisms is built on the category of partial morphisms of event structures described in [CCRW17]. We first recall:

Definition 3.6. A *partial morphism* of event structures is a partial function $f : E \rightarrow F$ such that

- (i) preserves configurations: $\forall x \in \mathcal{C}(E), f(x) \in \mathcal{C}(F)$;
- (ii) is *locally injective*: $\forall e, e' \in x \in \mathcal{C}(E)$, if $f(e) = f(e')$ (both defined) then $e = e'$.

For partial Σ -morphisms, changes with definition 3.2 is slightly bigger since knowledge preservation is not defined without Σ -independence:

Definition 3.7. A *partial Σ -morphism* is a partial morphism of event structures $f : E \rightarrow F$ such that the domain $V \subseteq E$ of f is Σ -independent and for all $x \in \mathcal{C}(E)$, $\lambda_F^{f(x)} \preceq f \cdot \lambda_E^x$.

Note that for every Σ -event structure E and every of its Σ -independent subset $V \subseteq E$, there is a partial Σ -morphism $\mathfrak{h} : E \rightarrow E \downarrow V$ that is defined as the identity on V .

Proposition 3.3. Σ -event structures and their partial morphisms define a monoidal category $(\mathcal{E}_{\Sigma, \perp}, \parallel, \emptyset)$.

Proof. Very similar to \mathcal{E}_{Σ} , it is routine to check that partial morphisms of event structures compose and have identities as neutrals. The monoidal structure of (\parallel, \emptyset) is orthogonal to having partial maps, except additionally, \emptyset becomes terminal. \square

Total Σ -morphisms still define morphisms in $\mathcal{E}_{\Sigma, \perp}$. Partial and total Σ -morphisms then relate in the following way:

Proposition 3.4. Let $f : E \rightarrow F$ be a partial Σ -morphism with domain $V \subseteq E$. Then f decomposes into $f_{\downarrow V} \circ \mathfrak{h} : E \rightarrow E \downarrow V \rightarrow F$ and for every other partial-total factorisation $h \circ f' : E \rightarrow X \rightarrow G$ of f , there exists a unique total map $h' : E \downarrow V \rightarrow X$ such that

$$\begin{array}{ccccc}
 & E & & & \\
 & \downarrow \mathfrak{h} & \searrow h & & \\
 f \swarrow & E \downarrow V & \xrightarrow{h'} & X & \\
 & \downarrow f_{\downarrow V} & \swarrow f' & & \\
 & G & & &
 \end{array}$$

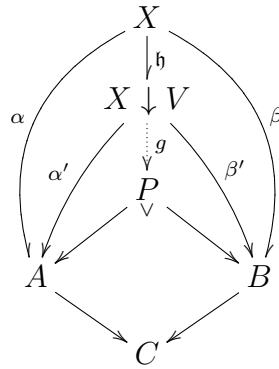
We say that \mathfrak{h} has the partial-total universal property and call it a hiding map.

Proof. All f, \mathfrak{h}, h have domain V , so following the definition of \mathfrak{h} , $h' = h_{\upharpoonright V}$ is the only possible morphism that makes the above diagram commutes. \square

Although pullbacks are not defined in general for partial Σ -morphisms, $\mathcal{E}_{\Sigma, \perp}$ inherits the ones from \mathcal{E}_{Σ} :

Proposition 3.5. *Pullbacks of total Σ -morphisms still define pullbacks in $\mathcal{E}_{\Sigma, \perp}$.*

Proof. This observation also holds in the plain case and a detailed proof for it is given in [CCRW17]. As this proof only relies on the universal property of pullbacks we do not reproduce it her but summarise it by the following diagram – where V denotes the domain of α and β .



Uniqueness follows by uniqueness of g and the universal property of \mathfrak{h} . \square

Zippering lemma The rest of this section is dedicated to prove the following zippering lemma, in fact it is almost a restatement of what is done in [CCRW17].

Lemma 3.6 (Zippering Lemma). *Let $\sigma : S \rightarrow A \parallel B \parallel C$, $\sigma' : S' \rightarrow A \parallel C$ and $\rho : U \rightarrow C \parallel D$ be Σ -morphisms and let $\mathfrak{h} : S \rightarrow S'$ be a hiding map making the following diagram commutes:*

$$\begin{array}{ccc} S & \xrightarrow{\mathfrak{h}} & S' \\ \sigma \downarrow & & \downarrow \sigma' \\ A \parallel B \parallel C & \xrightarrow{A \parallel \perp \parallel C} & A \parallel C \end{array}$$

Then, there is a Σ -morphism $U \otimes \mathfrak{h} : U \otimes S \rightarrow U \otimes S'$ that is a hiding map.

The existence of $U \circledast \mathfrak{h}$ is immediate following the pullback property of $U \circledast S$ and proposition 3.5.

$$\begin{array}{ccccc}
 & U \circledast S & \xrightarrow{U \circledast \mathfrak{h}} & U \circledast S' & \\
 & \downarrow & & \downarrow & \\
 S \parallel D & \xrightarrow{\mathfrak{h} \parallel D} & A \parallel B \parallel U & \xrightarrow{A \parallel \perp \parallel U} & A \parallel U \\
 \downarrow \sigma \parallel D & \swarrow A \parallel B \parallel \rho & & \searrow \sigma' \parallel D & \downarrow A \parallel \rho \\
 A \parallel B \parallel C \parallel D & \xrightarrow{A \parallel \perp \parallel C \parallel D} & & & A \parallel C \parallel D
 \end{array}$$

We show that it is a hiding morphism using the following characterisation of partial-total universal property:

Proposition 3.6. *Let $f : E \rightarrow F$ be a partial Σ -morphism defined on $V \subset E$, the followings assertions are equivalent:*

- (i) *f has the partial-total universal property*
- (ii) *There exists an isomorphism $\varphi : E \downarrow V \cong F$ such that $\varphi \circ \mathfrak{h}_V = f$ (in fact $\varphi = f \upharpoonright_V$)*
- (iii) *f has a knowledge-preserving hiding witness, that is, a monotonic function $\text{wit}_f : \mathcal{C}(F) \rightarrow \mathcal{C}(E)$ such that for every $x \in \mathcal{C}(E)$, $\text{wit}_f(f(x)) \subseteq x$ and, for every $y \in \mathcal{C}(F)$, $f(\text{wit}_f(y)) = y$ and $f.\lambda_E^{\text{wit}_f(y)} \preceq \lambda_F^y$.*

Proof. Removing the knowledge preserving condition in (iii) the same proposition holds in \mathcal{E}_\perp . Our proof is thus heavily inspired from the one in [CCRW17].

(i) \Leftrightarrow (ii). From right to left, by proposition 3.4, \mathfrak{h}_V has partial-total universal which transports through φ . From left to right, both $\mathfrak{h}_V : E \rightarrow E \downarrow V$ and $f : E \rightarrow F$ have partial-total universal property, yielding the desired isomorphism.

(i) \Rightarrow (iii). Using the above equivalence, it is enough to show that this implication holds for $\mathfrak{h}_V : E \rightarrow E \downarrow V$. For $x \in \mathcal{C}(E \downarrow V)$, define $\text{wit}(x) = [x]_E \in \mathcal{C}(E)$. Clearly, $\mathfrak{h}(\text{wit}(x)) = [x] \cap V = x$ and $\text{wit}(\mathfrak{h}(x)) = [x \cap V] \subseteq x$ as required. It is monotonic and it preserves terms by definition.

(iii) \Rightarrow (ii). Following lemma 1.4 we construct an order-isomorphism on configurations:

$$\begin{array}{lcl}
 p & : & \mathcal{C}(E \downarrow V) \rightarrow \mathcal{C}(F) \\
 & & x \mapsto f([x]_E)
 \end{array}$$

$$\begin{array}{lcl}
 q & : & \mathcal{C}(F) \rightarrow \mathcal{C}(E \downarrow V) \\
 & & y \mapsto \text{wit}(y) \cap V
 \end{array}$$

It is clear by definition that these maps are monotonic, we need to prove that they are inverse of each other. For one direction, let $y \in \mathcal{C}(F)$, then $\text{wit}(y)$ is down-closed in E (since $\text{wit}(y) \in \mathcal{C}(E)$) and thus can only differ from $[\text{wit}(y) \cap V]_E$ by events not in V . Hence, $f([\text{wit}(y) \cap V]_E) = f(\text{wit}(y)) = y$, *i.e.* $p \circ q(y) = y$.

For the other direction, we note first that if $x \in \mathcal{C}(E)$ has all its maximal events in V , then $\text{wit}(f(x)) = x$. Indeed, $\text{wit}(f(x)) \subseteq x$ by hypothesis and both sides map to $f(x)$ via f , so, by local injectivity, $\text{wit}(f(x)) \cap V = x \cap V$. But $x = [x \cap V]_E$ since its maximal elements are visible. So, putting everything together:

$$x = [x \cap V] = [\text{wit}(f(x)) \cap V] \subseteq \text{wit}(f(x)) \subseteq x$$

Now, let $x \in \mathcal{C}(E \downarrow V)$, then, by definition, $[x]$ has its maximal events in V , so $\text{wit}(f([x])) = [x]$, and so $q \circ p(x) = \text{wit}(f([x])) \cap V = [x] \cap V = x$. This concludes the other direction.

By Lemma 1.4, p and q yield isomorphisms $\hat{p} = f|_V : E \downarrow V \rightarrow F$ and $\hat{q} : F \rightarrow E \downarrow V$ in \mathcal{E} . It is routine to check that these isomorphisms belong to \mathcal{E}_Σ as well:

- let $x \in \mathcal{C}(E \downarrow V)$, then $\lambda_F^{\hat{p}(x)} = \lambda_F^{f([x])} \preceq f.\lambda E^x = f|_V.\lambda_{E \downarrow V}^x = \hat{p}.\lambda_{E \downarrow V}^x$.
- let $y \in \mathcal{C}(F)$ then $\hat{p}.\lambda_{E \downarrow V}^{\hat{q}(y)} = f|_V.\lambda_{E \downarrow V}^{\text{wit}(y) \cap V} = f.\lambda_E^{\text{wit}(y)} \preceq \lambda_F^y$, hence, by renaming, $\lambda_{E \downarrow V}^{\hat{q}(y)} \preceq \hat{q}.\lambda_F^y$.

□

Using the above characterisation, we conclude the proof of the zipping lemma by constructing a term-preserving hiding witness for $U \otimes \mathfrak{h}$, using the hiding witness of \mathfrak{h} .

Lemma 3.7. *$U \otimes \mathfrak{h}$ has a knowledge-preserving hiding witness:*

$$\begin{aligned} \text{wit} : \mathcal{C}(U \otimes S') &\rightarrow \mathcal{C}(U \otimes S) \\ x_U \otimes x_{S'} &\mapsto x_U \otimes \text{wit}_{\mathfrak{h}}(x_{S'}) \end{aligned}$$

Proof. From the zipping lemma in [CCRW17], we know that the above map defines a hiding witness so we just have to check that it is knowledge preserving.

For that we prove the following: given $x_U \otimes x_{S'} \in \mathcal{C}(U \otimes S')$, $\lambda_{U \otimes S'}^{x_U \otimes x_{S'}}$ can be used to construct a unifier μ for $\mathcal{U}_{\varphi_{x_U \otimes \text{wit}(x_{S'})}}$ such that $(U \otimes \mathfrak{h}).\mu = \lambda_{U \otimes S'}^{x_U \otimes x_{S'}}$. Hence, by lemma 2.2 and the minimality of $\lambda_{U \otimes S}^{x_U \otimes \text{wit}(x_{S'})}$ we have $\lambda_{U \otimes S}^{x_U \otimes \text{wit}(x_{S'})} \preceq \mu$ and so $(U \otimes \mathfrak{h}).\lambda_{U \otimes S}^{x_U \otimes \text{wit}(x_{S'})} \preceq (U \otimes \mathfrak{h}).\mu = \lambda_{U \otimes S'}^{x_U \otimes x_{S'}}$.

Let us construct μ . First note that $\mathcal{U}_{x_U \otimes \text{wit}(x_{S'})}$ can be split into $\mathcal{U}_{x_U \otimes x_{S'}}[(U \otimes \mathfrak{h})^{-1}]$ and $\{\lambda_S^{\text{wit}(x_{S'})}(s)[\pi_1^{-1}] \doteq \lambda_B^{x_B}(\sigma(s))[\pi_2^{-1}]\}_{s \in \text{wit}(x_{S'})_B}$, where $\text{wit}(x_{S'})_B$ is the set of events in $x_{S'}$ that maps to B via σ and $\sigma(\text{wit}(x_{S'})_B) = x_B$. These two sub-problems are independently solved by $(U \otimes \mathfrak{h})^{-1} \cdot \lambda_{U \otimes S'}^{x_U \otimes x_{S'}}$ and $\pi_1^{-1} \cdot \lambda_{S|\text{wit}(x_{S'})_B}^{\text{wit}(x_{S'})}$. Hence, taking their composition gives a unifier for the general problem $\mathcal{U}_{x_U \otimes \text{wit}(x_{S'})}$ and, by definition, its projection via $U \otimes \mathfrak{h}$ is equal to $\lambda_{U \otimes S'}^{x_U \otimes x_{S'}}$. \square

Besides the zipping lemma, proposition 3.6 leads to a couple of useful corollaries.

Corollary 3.1. *Let $\mathfrak{h}_1 : E_1 \rightarrow E_2$ and $\mathfrak{h}_2 : E_2 \rightarrow E_3$ be two hiding morphisms, then $\mathfrak{h}_2 \circ \mathfrak{h}_1 : E_1 \rightarrow E_3$ is a hiding map.*

Proof. By composition of knowledge-preserving hiding witnesses from characterisation (iii) of hiding morphisms. \square

Corollary 3.2. *Let $\mathfrak{h}_1 : E_1 \rightarrow E_2$ and $\mathfrak{h}_2 : E_3 \rightarrow E_4$ be two hiding morphisms, then $\mathfrak{h}_1 \parallel \mathfrak{h}_2 : E_1 \parallel E_3 \rightarrow E_2 \parallel E_4$ is a hiding map.*

Proof. By composition of knowledge-preserving hiding witnesses from characterisation (iii) of hiding morphisms. \square

3.2 Compact closed structure of Σ -CG

The previous section introduces $\mathcal{E}_{\Sigma, \perp}$, a categorical framework to talk about interaction and projection of morphisms of event structures with annotations. Similarly to what is done in [CCRW17] for the plain case, we now show that certain total morphisms of $\mathcal{E}_{\Sigma, \perp}$ can be viewed as *strategies* and that pullback and hiding in $\mathcal{E}_{\Sigma, \perp}$ provide a basis for studying properties of their composition. This helps concluding on the compact closed structure of Σ -CG.

3.2.1 From Σ -morphisms to Σ -strategies

Games and Σ -strategies as defined in chapter 2.1 can be viewed as particular Σ -event structures and morphisms. To characterize them inside Σ -ES we introduce polarities back.

Definition 3.8. A Σ -event structure with polarities is a Σ -event structure (E, λ_E) together with a *polarity function* $\text{pol}_E : E \rightarrow \{+, -\}$.

The dual operator $(-)^{\perp}$ is still defined by reversing polarities. In fact, adding polarities is orthogonal to adding annotations and it is direct to show:

Proposition 3.7. *Σ -event structure with polarities and their polarity preserving Σ -morphisms define a category \mathcal{EP}_Σ , equipped with a monoidal structure (\parallel, \emptyset) .*

The same holds with partial morphisms, yielding a category $\mathcal{EP}_{\Sigma, \perp}$.

The functors \mathcal{F} and \mathcal{G} between \mathcal{E} and \mathcal{E}_Σ , described on page 74, are still define for \mathcal{EP} and \mathcal{EP}_Σ (functors now preserve polarities). We call *games* in \mathcal{EP}_Σ the images of \mathcal{EP} via \mathcal{F} .

As mentioned in chapter 1, morphisms of the form $\sigma : S \rightarrow A^\perp \parallel B$ in \mathcal{EP} can be viewed as *pre-strategies*, then denoted $\sigma : A \dashv\vdash B$. This is because one can define composition \odot on them – as interaction followed by projection. Having defined interactions and projections in \mathcal{EP}_Σ it is tempting to view Σ -morphisms of that form as pre-strategies as well. Yet, composition in that case may not always be defined as projection in \mathcal{EP}_Σ is subject to restriction. Using polarities we give a criterion that restricts the shape of Σ -morphisms and helps recovering a notion of pre-strategies.

Definition 3.9. A morphism of Σ -event structure with polarities $\sigma : S \rightarrow A$ is *productive* if for every $x \in \mathcal{C}(S)$, $\text{fv}(x) \subseteq x^-$.

This definition is based on the observation that splitting the set of variables that can be used freely between σ and τ is enough to avoid unspecified variables from the common game at the end of interaction. A natural and homogeneous way to rule this split between σ and τ is to specify this splitting it using their common game, for example exploiting polarities.

Proposition 3.8. *If $\sigma : S \rightarrow A^\perp \parallel B$ and $\tau : T \rightarrow B^\perp \parallel C$ are productive Σ -morphisms, then their composition $\tau \odot \sigma : T \odot S \rightarrow A^\perp \parallel C$ is well-defined in \mathcal{EP}_Σ .*

Proof. One needs to check that the set of visible events in $T \otimes S$ is Σ -independent. This is a consequence of the fact that for $\varphi : x_S \parallel x_C \simeq x_A \parallel x_T$, a Σ -secured bijection, $\pi_1^{-1} \lambda_{S \parallel C}^{x_S \parallel x_C}[\lambda^\varphi] = \lambda^\varphi = \pi_2^{-1} \lambda_{A \parallel T}^{x_A \parallel x_T}[\lambda^\varphi]$ (corollary 2.1) hence

$$\begin{aligned} \text{fv}(\lambda^\varphi) \subseteq & \pi_1^{-1}(\text{fv}(\lambda_S^{x_S})) \cap \pi_2^{-1}(\text{fv}(\lambda_T^{x_T})) \\ & \cup \pi_1^{-1}(\text{fv}(\lambda_S^{x_S})) \cap \pi_2^{-1}(\text{fv}(\lambda_A^{x_A})) \\ & \cup \pi_1^{-1}(\text{fv}(\lambda_C^{x_C})) \cap \pi_2^{-1}(\text{fv}(\lambda_T^{x_T})) \end{aligned}$$

By productivity, the first intersection is empty and by definition of Σ -morphisms the last two are respectively included in $\pi_2^{-1}(A^\perp)$ and $\pi_1^{-1}(C)$, concluding that

$$\text{fv}(\lambda^\varphi) \subseteq \tau \odot \sigma(A \parallel C)$$

□

Other constructions on pre-strategies in \mathcal{EP} are defined in \mathcal{EP}_Σ . Reusing notation, the tensor product of two games is $A \otimes B = A \parallel B$ and for two Σ -morphisms $\sigma : A \rightarrow B$, $\tau : C \rightarrow D$

$$\sigma \otimes \tau = \gamma \circ (\sigma \parallel \tau)$$

with $\gamma : A^\perp \parallel C \parallel B^\perp \parallel D \rightarrow A^\perp \parallel B^\perp \parallel C \parallel D$ the corresponding isomorphism in \mathcal{EP}_Σ . Following definitions 1.14 and 1.9 from chapter 1 one similarly define *left and right renaming* via Σ -isomorphisms as well as *lifting and co-lifting* of Σ -isomorphisms. Note that these constructions preserve productivity.

Finally we extend the definition for isomorphic pre-strategies (definition 1.8), we have:

Definition 3.10. Two parallel Σ -morphisms $\sigma_1, \sigma_2 : A \rightarrow B$ are *isomorphic* if there exists an isomorphism of Σ -event structures that makes the triangle below commutes

$$\begin{array}{ccc} & \phi & \\ & \curvearrowright & \\ S_1 & & S_2 \\ & \sigma_1 \searrow & \swarrow \sigma_2 \\ & A^\perp \parallel B & \end{array}$$

Note that in general, for $x \in \mathcal{C}(S_1)$, $\phi \cdot \lambda_{S_1}^x \neq \lambda_{S_2}^{\varphi(x)}$; these two annotations are only α -equivalent. However, when restricted to the Σ -morphisms that will represent the Σ -strategies in Σ -ES, we will get this equality back, thus matching definition 2.4

Having a notion of composition, tensor product and isomorphism for Σ -pre-strategies – that are productive Σ -morphisms – we now bridge the definition of Σ -strategies given in chapter 2.1 with the one of Σ -morphisms.

Definition 3.11. Let $(\sigma, \lambda_\sigma) : A \xrightarrow{\Sigma} B$ be a Σ -strategy of support S , then σ defines a Σ -pre-strategy

$$\sigma^\lambda : (S, \lambda_\sigma) \rightarrow \mathcal{F}(A^\perp \parallel B)$$

with λ_σ viewed as a family of annotations, defined for $x \in \mathcal{C}(S)$ by $\lambda_{\sigma|x^+}^x = \lambda_{\sigma|x^+}$ and $\lambda_{\sigma|x^-}^x = \text{id}_{x^-}$.

Note that for (σ, λ_σ) a Σ -strategy, $\lambda_\sigma(s) = \lambda_{\sigma^\lambda}^{[s]}(s)$ and for every $x \subseteq x' \in \mathcal{C}(S)$, $\lambda_\sigma^x = \lambda_{\sigma|x}$.

Σ -strategies and their Σ -morphism counterparts really are two different views of the same object (a meager view and a fat view), in particular:

Proposition 3.9. *Let $\sigma, \sigma' : A \xrightarrow{\Sigma} B$, $\tau : B \xrightarrow{\Sigma} C$, $\rho : D \rightarrow E$ be Σ -strategies, let $f : B \cong B'$ be a game isomorphism then*

1. $(\sigma \odot \tau)^\lambda = \sigma^\lambda \odot \tau^\lambda$;
2. $\phi : \sigma_1 \simeq \sigma_2$ iff $\phi : \sigma_1^\lambda \simeq \sigma_2^\lambda$;
3. $\sigma^\lambda \otimes \rho^\lambda = (\sigma \otimes \rho)^\lambda r$;
4. $(f \cdot \tau)^\lambda = f \cdot \tau^\lambda$.

Proof. 1. Consider a secured bijection $x_T \otimes x_S \in \mathcal{C}(T \otimes S)$ then for every $(s, t) \in x_T \otimes x_S$ its corresponding equation in $\mathcal{U}_{x_T \otimes x_S}$ is of the form

$$\begin{cases} \lambda_\sigma(s)[\pi_1^{-1}] & \doteq & s[\pi_1^{-1}] & \text{if } s \in x_S^+ \\ t[\pi_2^{-1}] & \doteq & \lambda_\tau(t)[\pi_2^{-1}] & \text{if } t \in x_T^+ \\ (s, t) & \doteq & (s, t) & \text{otherwise} \end{cases}$$

By theorem 2.2, $\mathcal{U}_{x_T \otimes x_S}$ has a solution and $\lambda_{\tau \otimes \sigma}$ as described in lemma 2.3 is exactly the maximum substitution from corollary 2.1. Hence the equality.

2. Left to right implication is immediate from definition 2.4 as $\phi \cdot \lambda_{\sigma_1} = \lambda_{\sigma_2}$. Let's focus on its contrapositive. Condition for isomorphisms in \mathcal{EP}_Σ are more lax than in CG, yet the specificities of Σ -strategy images make the two definitions equivalent. Let $s_1 \in S_1$ and $s_2 = \phi(s_1)$, then from lemma 3.1 $\phi \cdot \lambda_{\sigma_1}^{[s_1]} = \lambda_{\sigma_2}^{[s_2]}[\phi \cdot \lambda_{\sigma_1}^{[s_1]}]$. By definition $\text{fv}(\lambda_{\sigma_2}^{[s_2]}) \subseteq [s_2]^-$ and $(\phi \cdot \lambda_{\sigma_1}^{[s_1]})_{|[s_2]^-} = \text{id}_{[s_2]^-}$ hence the previous equality simplifies to $\phi \cdot \lambda_{\sigma_1}^{[s_1]} = \lambda_{\sigma_2}^{[s_2]}$ as desired.

3,4. This is immediate by definition of \otimes and \cdot in both Σ -CG and \mathcal{EP}_Σ . \square

3.2.2 Σ -CG: a compact closed category

We use the above correspondence to prove associativity for composition, functoriality of tensor product on Σ -strategies as well as the annotated version of proposition 3.12 relating renaming with composition. Following the proof scheme described in section 1.3 for the plain case, this completes propositions 2.2 and 2.3 in showing that CG_Σ is a symmetric monoidal category. Following [CCRW17], our proof technique relies on the pullback and hiding universal properties of interaction and projection in $\mathcal{EP}_{\Sigma, \perp}$.

Associativity of \odot Given three Σ -pre-strategies $\sigma : S \rightarrow A^\perp \parallel B$, $\tau : T \rightarrow B^\perp \parallel C$, $\rho : U \rightarrow C^\perp \parallel D$, by pullback property, their interaction is associative:

$$\begin{array}{c}
 U \otimes (T \otimes S) \xleftarrow{\sim} (U \otimes T) \otimes S \\
 \swarrow \quad \searrow \\
 (T \otimes S) \parallel D \quad (U \otimes T) \otimes S \\
 \swarrow \quad \searrow \quad \downarrow \quad \swarrow \quad \searrow \\
 (S \parallel C) \parallel D \quad (A \parallel T) \parallel D \quad (A \parallel B) \parallel U \quad S \parallel (C \parallel D) \quad A \parallel (T \parallel D) \quad A \parallel (B \parallel U) \\
 \swarrow \quad \searrow \quad \downarrow \quad \swarrow \quad \searrow \quad \downarrow \quad \swarrow \quad \searrow \\
 (\sigma \parallel C) \parallel D \quad (A \parallel B \parallel C) \parallel D \quad (A \parallel B) \parallel \rho \quad \sigma \parallel (C \parallel D) \quad A \parallel (B \parallel C \parallel D) \quad A \parallel (B \parallel \rho)
 \end{array}$$

The corresponding isomorphism of Σ -event structure is denoted $a_{\sigma,\tau,\rho} : U \otimes (T \otimes S) \cong (U \otimes T) \otimes S$. Because of the partial-total universal properties of projection this isomorphism projects down to composition.

Proposition 3.10. *Let $\sigma : S \rightarrow A^\perp \parallel B$, $\tau : T \rightarrow B^\perp \parallel C$, $\rho : U \rightarrow C^\perp \parallel D$ be three Σ -pre-strategies then $\rho \odot (\tau \odot \sigma) \simeq (\rho \odot \tau) \odot \sigma$.*

Proof. Instantiating the zipping lemma (3.6) with hiding maps from the composition of Σ -pre-strategies, we first define the following two partial Σ -morphisms:

$$\mathfrak{h}_{(\sigma,\tau),\rho} : U \otimes (T \otimes S) \xrightarrow{U \otimes \mathfrak{h}} U \otimes (T \odot S) \xrightarrow{\mathfrak{h}} U \odot (T \odot S)$$

$$\mathfrak{h}_{\sigma,(\tau,\rho)} : (U \otimes T) \otimes S \xrightarrow{\mathfrak{h} \otimes S} (U \odot T) \otimes S \xrightarrow{\mathfrak{h}} (U \odot T) \odot S$$

By composition of hiding maps (lemma 3.1), these two morphisms are hiding maps. Moreover, ignoring the dotted arrow, they make the diagram below to commute:

$$\begin{array}{ccc}
 U \otimes (T \otimes S) & \xrightarrow{\sim} & (U \otimes T) \otimes S \\
 \mathfrak{h}_{\sigma,(\tau,\rho)} \downarrow & & \downarrow \mathfrak{h}_{(\sigma,\tau),\rho} \\
 U \odot (T \odot S) & \xrightarrow{\sim} & (U \odot T) \odot S \\
 \rho \odot (\tau \odot \sigma) \swarrow & & \swarrow (\rho \odot \tau) \odot \sigma \\
 & A \parallel D &
 \end{array}$$

Thus, by the partial-total universal property there exists $\alpha_{\sigma,\tau,\rho} : U \odot (T \odot S) \cong (U \odot T) \odot S$ that makes the above diagram commutes. \square

Viewing Σ -strategies through their Σ -morphisms counterparts as described in proposition 3.9, the proposition above provides a proof of the associativity of composition for Σ -strategies as stated in proposition 2.1.

Functoriality of \otimes . In proposition 2.3 we proved that the tensor operation on Σ -strategies preserves identities. We now complete the proof that it is a bifunctor, showing that it preserves composition as well. Again, this relies on the composition of Σ -strategies being described as a pullback plus hiding in $\mathcal{E}_{\Sigma, \perp}$.

Proposition 3.11. *Let*

$$\begin{array}{ll} \sigma_1 : S_1 \rightarrow A_1^\perp \parallel B_1 & \tau_1 : T_1 \rightarrow B_1^\perp \parallel C_1 \\ \sigma_2 : S_2 \rightarrow A_2^\perp \parallel B_2 & \tau_2 : T_2 \rightarrow B_2^\perp \parallel C_2 \end{array}$$

be Σ -pre-strategies, then,

$$(\tau_1 \odot \sigma_1) \otimes (\tau_2 \odot \sigma_2) \simeq (\tau_1 \otimes \tau_2) \odot (\sigma_1 \otimes \sigma_2)$$

Proof. This follows the same proof scheme as the one for the plain case found in Proposition 5.2 of [CCRW17].

First note that the parallel composition of pullbacks still defines a pullback. Hence, by pullback property there is an isomorphism $(T_1 \otimes S_1) \parallel (T_2 \otimes S_2) \cong (T_1 \parallel T_2) \otimes (S_1 \parallel S_2)$ such that the following diagram commutes (the isomorphism is the dotted arrow)

$$\begin{array}{ccccc} & & (T_1 \otimes S_1) \parallel (T_2 \otimes S_2) & & \\ & \searrow^{\mathfrak{h} \parallel \mathfrak{h}} & \downarrow^{(\tau_1 \otimes \sigma_2) \parallel (\tau_1 \otimes \sigma_2)} & & \swarrow^{\text{dotted}} \\ (T_1 \odot S_1) \parallel (T_2 \odot S_2) & & (A_1 \parallel B_1 \parallel C_1) \parallel (A_2 \parallel B_2 \parallel C_2) & & (T_1 \parallel T_2) \otimes (S_1 \parallel S_2) \\ \downarrow^{(\tau_1 \odot \sigma_2) \parallel (\tau_1 \odot \sigma_2)} & \searrow^{\mathfrak{h}} & \downarrow^{\gamma} & \swarrow^{(\tau_1 \otimes \tau_2) \otimes (\sigma_1 \otimes \sigma_2)} & \downarrow^{\mathfrak{h}} \\ (A_1 \parallel C_1) \parallel (A_2 \parallel C_2) & & (A_1 \parallel A_2) \parallel (B_1 \parallel B_2) \parallel (C_1 \parallel C_2) & & (T_1 \parallel T_2) \odot (S_1 \parallel S_2) \\ & \searrow^{\gamma} & \downarrow^{\mathfrak{h}} & \swarrow^{(\tau_1 \otimes \tau_2) \odot (\sigma_1 \otimes \sigma_2)} & \\ & & (A_1 \parallel A_2) \parallel (C_1 \parallel C_2) & & \end{array}$$

The rest of the diagram commutes by definition. Moreover, the parallel composition of hiding maps define a hiding map (corollary 3.2) so by partial-total universal property there is an isomorphism such that (dotted arrow)

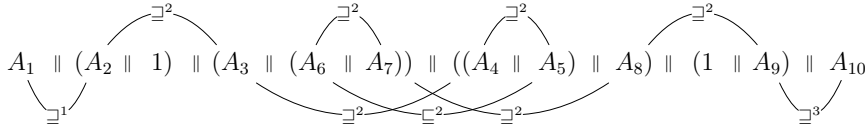
$$\begin{array}{ccc} (T_1 \odot S_1) \parallel (T_2 \odot S_2) & \text{dotted} & (T_1 \parallel T_2) \odot (S_1 \parallel S_2) \\ \downarrow^{(\tau_1 \odot \sigma_2) \parallel (\tau_1 \odot \sigma_2)} & & \downarrow^{(\tau_1 \otimes \tau_2) \odot (\sigma_1 \otimes \sigma_2)} \\ (A_1 \parallel C_1) \parallel (A_2 \parallel C_2) & \xrightarrow{\gamma} & (A_1 \parallel A_2) \parallel (C_1 \parallel C_2) \end{array}$$

which validates the equation of the proposition. \square

Let us focus on the first one, and let us consider them as Σ -morphisms. By the zipping Lemma 3.6, there is a hiding map from the 5-ary interaction to the resulting right hand-side composition that commutes with the projection. The base game of the interaction is

$$A \parallel (A \parallel 1) \parallel (A \parallel (A^\perp \parallel A)) \parallel ((A \parallel A^\perp) \parallel A) \parallel (1 \parallel A) \parallel A$$

Every strategy involved in the interaction add constraints on configurations that do not conflict with each other and that are subsumed by:



Hence configurations of the interaction correspond to the data of 10 configurations x_1, \dots, x_{10} of A that satisfy the above constraints (with indices reflecting the numbering above). This is equivalent to saying that for every $a \in x_1$ with $\text{pol}_A(a) = +$, there exists an index $k_a \geq 1$ such that $a \in x_i$ if $i \leq k$ and $a \notin x_j$ if $j > k$. And symmetrically for $a \in x_{10}$, $\text{pol}_A(a) = -$, \geq and $<$. Moreover, the annotations of the strategies, follow exactly these causal constraints. In other words, every configuration generates a unification problem consisted in:

$$\bigcup_{\substack{a_1 \in x_1 \\ \text{pol}_A(a_1) = -}} \left\{ \begin{array}{c} a_1 \doteq a_1 \\ a_2 \doteq a_1 \\ \dots \\ a_{k_a} \doteq a_{k_a-1} \end{array} \right\} \quad \bigcup_{\substack{a_{10} \in x_{10} \\ \text{pol}_A(a_{10}) = +}} \left\{ \begin{array}{c} a_{10} \doteq a_{10} \\ a_9 \doteq a_{10} \\ \dots \\ a_{k_a} \doteq a_{k_a+1} \end{array} \right\}$$

where a_i denotes the occurrence of an event $a \in A$ in the i^{th} component of the configuration.

In the above characterisation, configurations with all their maximal events visible correspond exactly to configurations of \mathbb{C}_A (after hiding of the intermediate components). Moreover, the resulting labelling (restricted to outer component) is exactly the one of $\lambda_{\mathbb{C}_A}$. \square

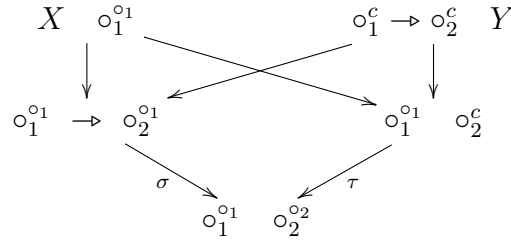
This concludes the proof that CG_Σ (quotiented by \simeq) forms a compact closed category.

3.3 Observations and remarks

This chapter introduced a category of Σ -event structure as a more general framework to reason about Σ -strategies. The definition of annotations on

Σ -event structures is more general than on Σ -strategies as the annotations may vary with configurations. The first reason for this generalisation is that if annotations on event structures are restricted to events – or to say it otherwise, to annotations on configurations such that knowledge preservation actually is an identity – then the category has no pullbacks in general.

Example 3.1. Consider the Σ -morphisms σ and τ depicted below:



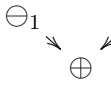
If the annotations on Σ -event structures were to be fixed once and for all on every event and could not vary with configurations, then these two morphisms would not have a pullback. Indeed, no Σ -event structure with annotations defined on events can subsume *both* the annotations of X and the annotations of Y . The only way for that is to be allowed to change annotations in between configuration $\{\circ_1\}$ and configuration $\{\circ_1, \circ_2\}$ – as does the actual pullback in \mathcal{E}_Σ defined by $\circ_1 \rightarrow \circ_2$ with annotated configurations $\{\circ_1^{\circ_1}\}$ and $\{\circ_1^c \rightarrow \circ_2^c\}$.

For the particular kind of Σ -morphisms – *i.e.* the Σ -strategies – we actually exploit in $\Sigma\text{-}\mathcal{E}$, the above generality is not required: as reflected in the definition of the interaction for Σ -strategies 2.3, Σ -morphisms that arose from Σ -strategies do have pullbacks that can be described using annotations on events only.

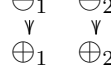
Instead of looking for general pullbacks, we could thus have directly restricted our study to Σ -strategies in $\Sigma\text{-}\mathcal{E}$; this would have been enough for the applications we consider in part II and III for which annotations are simple witnesses, or book-keeper: they do not interfere with what is happening in the strategies at the configuration level and are constrained by the causal dependencies on events.

Despite being heavier than the event based annotations, we believe that the more general annotations on configurations can have an interest for modelling settings where annotations actually do have an impact on the execution flow of strategies. As they stand, annotations on Σ -event structures does not explicitly generate new causal dependencies between events, however they may prevent some events from happening during an interaction. Exploiting the freeness of the general model we can derive the following patterns:

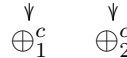
- Conditions: for $\ominus \rightarrow \oplus$ the annotations $\ominus^\ominus \mid \ominus^c \rightarrow \oplus^c$ imply that the move \oplus is only played during an interaction if \ominus was instantiated with some constant c .

- Conditional synchronisation: for $\ominus_1 \quad \ominus_2$ the annotations $\ominus_1^{\ominus_1} \quad \ominus_2^{\ominus_2}$

 (equal on every other subset) \mid $\ominus_1^{\ominus_1} \quad \ominus_2^{\ominus_1}$ imply that the move

\oplus is only played during an interaction if the two negative moves are instantiated with the same value.

- Conditional inconsistency: for $\ominus_1 \quad \ominus_2$ the annotations


$$\begin{array}{c|c} \begin{array}{c} \ominus_1^{\ominus_1} \quad \ominus_2^{\ominus_2} \\ \downarrow \quad \downarrow \\ \oplus_1^{\ominus_1} \end{array} & \begin{array}{c} \ominus_1^{\ominus_1} \quad \ominus_2^{\ominus_2} \\ \downarrow \quad \downarrow \\ \oplus_2^c \end{array} \end{array} \quad (\text{and their restrictions})$$

and $\ominus_1^{\ominus_1} \quad \ominus_2^{\ominus_1}$ imply that the two positive moves are inconsistent if


the two negative moves are not instantiated with the same value.

These patterns actually satisfy the productivity condition so Σ -pre-strategies can carry them. An other interesting question is to see whether they would be preserved by composition with copycat. It turns out they do.

More generally, one can put two wider conditions on the annotations of Σ -pre-strategies than simply be the image of a Σ -strategy in order to be preserved by composition with copycat:

- Σ -receptive: $\forall x \in \mathcal{C}(S), s \in (x - x)^*, \lambda_S^x(s) = s$;
- Σ -courteous: $\forall x \subseteq^- x' \in \mathcal{C}(S), \lambda_{x'}^x = \lambda_x$.

Coupled with receptivity, Σ -receptivity ensures that every Opponent move available in the game is allowed by a Player strategy, whatever its annotation from Opponent. Similarly to courtesy for causal dependencies, Σ -courtesy ensures, at the level of terms, that every change in the annotations is witnessed by a positive event. This allows for the two strategies to stay synchronised.

All the patterns listed above satisfy these two conditions. Moreover, these conditions are more general than the one for Σ -strategies, so in particular every structural morphisms ($\mathfrak{c}_A, \mathfrak{c}_\rho, \mathfrak{c}_s, \varepsilon, \dots$) validate them.

We believe that these conditions are preserved by composition, tensor and renaming and that they are sufficient (if not necessary, in the case of pre- Σ -strategies) for the composition with copycat to be neutral. However, proofs for these statements have not yet been written down.

If they do hold, then the constructions and proofs developed in this chapter will be general enough to show that the corresponding model defines a compact closed category. We leave this conjecture as future work, together with the exploration of models in which games themselves carry annotations, viewed as conditions on the annotations of strategies.

Simpler models

Chapters 2 and 3 present an enrichment of concurrent game models based on event structures by putting annotations on strategies. In part II and III, we make use of these enriched models to give a semantic for first order proofs and resource analysis of concurrent programs. In the first case, games and strategies are *deterministic*; in the second case, we deal with non determinism and conflicts, but we are not interested in remembering the exact branching or conflicting points in a strategy. In these contexts, the setting of concurrent games can be simplified leading to two lighter concurrent game models which inherit from the structural properties of T-CG.

In this chapter we introduce these simplified versions, named T-Det and T-Strat. These models can be viewed as enrichments of the simplified version of plain concurrent games as used by Castellan and Clairambault in [CC16]. They can also be viewed as an enrichment of Mellies and Mimram’s asynchronous games [MM07]. These models are closer to usual sequential games where strategies are defined as (set of) deterministic plays, except that concurrency is kept by having partial orders for plays instead of total ones.

In the first two sections, we recall the construction of Det and Strat, the lighter concurrent game models without annotations. There are no new results here but their explicit connections with CG is not detailed in the literature either. In the last section, we finally show that the previous connection between CG and its simpler models extends smoothly to T-annotations.

4.1 Games and rigid strategies

Rideau and Winskel’s concurrent games are suited to capture both concurrency and non-determinism/conflicts. A first simplification one can do on CG is to consider that strategies cannot internally distinguish two events if they map to the same move *with the same causal history*. In other words, these strategies validate the *non-deterministic idempotence* $a \vee a = a$ from processes. In concurrent games, this translate into

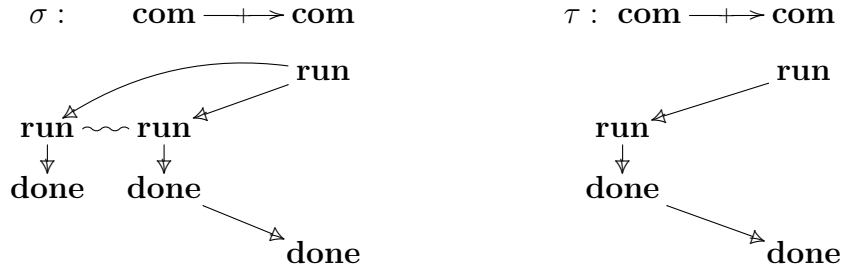


Figure 4.1: τ is non-deterministically idempotent but not σ .

Definition 4.1. A concurrent strategy $\sigma : S \rightarrow A$ is *non-deterministically idempotent* if for every $s_1, s_2 \in S$ such that $\sigma(s_1) = \sigma(s_2)$ and $[s_1]_S = [s_2]_S$ as partial order, then $s_1 = s_2$.

Note that if σ fails to be non-deterministically idempotent, then the faulty s_1, s_2 are necessarily in conflict, *i.e.* $\{s_1, s_2\} \notin \text{Con}_S$, as otherwise $[s_1] \cup [s_2]$ would be a configuration of S contradicting local injectivity.

For example, the left strategy on figure 4.1 does not validate non-deterministic idempotence as its two **run** moves are indistinguishable from a causal and naming point of view. In **Strat** – the simplified model of CG that we are introducing in this section –, this strategy will not be distinguishable from the strategy on the right of figure 4.1. Such distinction is useful to capture *e.g.* must convergence, but this is not under study in our next developments. Instead, we can focus on *rigid* strategies, that are non-deterministic idempotent strategies. As done in [CC16], we will see that in this case, one can simplify the setting of concurrent games by avoiding the usual move renaming: a strategy is described in terms of *augmentations*, that are, configurations of the game together with a partial order. Let us define the game model of rigid strategies **Strat** and show its connection with CG.

4.1.1 The compact closed structure of **Strat**

In **Strat**, games are still defined by event structures with polarities, and we keep the same definition of tensor and dual on them. What changes is the definition of strategies. Those are no longer described as morphisms of event structures but as sets of augmentations as defined in the next paragraph.

Rigid strategies First, we give the general notion of augmentations over a game A , they correspond to configuration $x_A \in \mathcal{C}(A)$ that is more causally constrained than (x_A, \leq_{x_A}) :

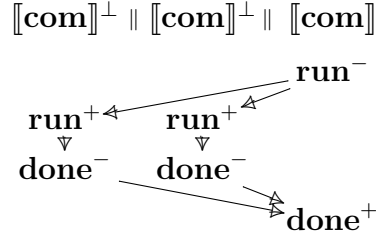


Figure 4.2: An augmentation for parallel composition

Definition 4.2. An *augmentation* over a game A is a partial order $\mathfrak{q} = (|\mathfrak{q}|, \leq_{\mathfrak{q}})$ such that $|\mathfrak{q}| \in \mathcal{C}(A)$ and \mathfrak{q} *respects the game*:

$$\forall a_1, a_2 \in |\mathfrak{q}|, \quad a_1 \leq_A a_2 \implies a_1 \leq_{\mathfrak{q}} a_2$$

We write $\text{Aug}(A)$ for the set of augmentations on A .

Note that polarities on A do not matter for this definition, we will sometimes use augmentations over event structures without polarities.

Example 4.1. Figure 4.2 displays an augmentation over the composed game $A = \llbracket \mathbf{com} \rrbracket^\perp \parallel \llbracket \mathbf{com} \rrbracket^\perp \parallel \llbracket \mathbf{com} \rrbracket$ where $\llbracket \mathbf{com} \rrbracket$ interprets the type of command for an imperative programming language, as introduced in example 2.3. This augmentation organises the events of A so that the two positive **run** arise in parallel after the negative **run** occurs while, the unique positive **done** event must wait for its two negative counterparts before happening. Later, we will see that this augmentation actually interprets the parallel command of the (affine) IPA programming language.

As partial orders are a special case of event structures, augmentations inherit all the operations on event structures and results stated in 1.1. In particular, augmentations can be rephrased as being partial orders such that $\mathcal{C}(\mathfrak{q}) \subseteq \mathcal{C}(A)$. Two augmentations are equal if their set of configurations are; inclusion of configurations also defines a partial order on $\text{Aug}(A)$:

Definition 4.3. Let $\mathfrak{q}, \mathfrak{q}' \in \text{Aug}(A)$ then $\mathfrak{q} \preceq \mathfrak{q}'$ iff $\mathcal{C}(\mathfrak{q}) \subseteq \mathcal{C}(\mathfrak{q}')$.

Rigid strategies are sets of augmentations that are closed under “prefix”. The adequate notion of prefix for augmentations is called *rigid inclusion*.

Definition 4.4. Let $\mathfrak{q}, \mathfrak{q}'$ be two partial orders. We say that \mathfrak{q} is *rigidly included* in \mathfrak{q}' , or that \mathfrak{q} is a *prefix* of \mathfrak{q}' , written $\mathfrak{q} \hookrightarrow \mathfrak{q}'$, if $|\mathfrak{q}| \in \mathcal{C}(\mathfrak{q}')$ and for every $a_1, a_2 \in |\mathfrak{q}|$,

$$a_1 \leq_{\mathfrak{q}} a_2 \iff a_1 \leq_{\mathfrak{q}'} a_2$$

We are now in position to define rigid strategies. By definition, an augmentation is a configuration of A that is more causally constrained than what is induced by \leq_A . As in CG, strategies are restricted in their choices of events and new causal constraints:

Definition 4.5. A *rigid strategy* on A , written $\sigma : A$, is a non-empty prefix-closed $\sigma \subseteq \text{Aug}(A)$, which additionally satisfies

- *receptivity*: for all $\mathfrak{q} \in \sigma$, if $|\mathfrak{q}| \xrightarrow{a^-} \text{C}$ in A , then $\mathfrak{q} \hookrightarrow \mathfrak{q}' \in \sigma$ such that $|\mathfrak{q}'| = |\mathfrak{q}| \cup \{a\}$;
- *courtesy*: for all $\mathfrak{q} \in \sigma$, if $a_1 \rightarrow_\sigma a_2$ and $\text{pol}_A(a_1, a_2) = (-, +)$, then $a_1 \rightarrow_A a_2$ as well (\mathfrak{q} is said to be *courteous*).

It follows by courtesy that \mathfrak{q}' is necessarily unique: the immediate dependency of a in \mathfrak{q}' is forced by its immediate dependency in A .

Example 4.2. A simple way to construct rigid strategies consists in taking the set of prefixes of a courteous augmentation $\mathfrak{q} \in \text{Aug}(A)$ that is *receptive*, i.e. such that if $|\mathfrak{q}| \xrightarrow{a} \text{C}$ in A , then $\text{pol}(a) = +$. For example, the set of prefixes of the augmentation depicted on figure 4.2 yields a rigid strategy.

An other way to construct rigid strategies is to derive them from concurrent strategies. Given a strategy $\sigma : S \rightarrow A$ in CG, every of its configurations $x \in \mathcal{C}(S)$ can be viewed as an augmentation $\mathfrak{q}_x = (\sigma(x), \leq_{\mathfrak{q}_x})$ where $\sigma(s_1) \leq_{\mathfrak{q}_x} \sigma(s_2)$ iff $s_1 \leq_S s_2$ – well defined as $\sigma : x \simeq \sigma(x)$ is a bijection. The augmentation \mathfrak{q}_x is called the *rigid image* of the configuration x . By courtesy of σ , \mathfrak{q}_x is courteous and the set

$$\text{Rig}(\sigma) = \{\mathfrak{q}_x \mid x \in \mathcal{C}(S)\}$$

is clearly closed under prefix and receptive, hence it defines a rigid strategy.

Example 4.3. The rigid version of copycat on A is given by

$$\text{Rig}(\mathfrak{c}_A) = \{\mathfrak{q}_{x\parallel y} \mid x, y \in A \text{ and } y \sqsubseteq x\} : A^\perp \parallel A$$

In the next section we will show that every rigid strategy is the image of some concurrent strategy and that the rigid image of copycat is neutral for the composition of rigid strategies that we now define.

Composition As previously, a rigid strategy defines a morphism from a game A to a game B if it plays on $A^\perp \parallel B$, also written $\sigma : A \xrightarrow{\text{Strat}} B$.

Composition of rigid strategies is defined component-wise by lifting composition of augmentations, which we shall first introduce.

Definition 4.6. Two dual augmentations $\mathfrak{q} \in \text{Aug}(A)$, $\mathfrak{q}' \in \text{Aug}(A^\perp)$ such that $|\mathfrak{q}| = |\mathfrak{q}'|$ are *causally compatible* if $(\leq_{\mathfrak{q}} \cup \leq_{\mathfrak{q}'})^*$ is a partial order. Then we write $\mathfrak{q} \wedge \mathfrak{q}' = (|\mathfrak{q}|, \leq_{\mathfrak{q} \wedge \mathfrak{q}'})$ for the resulting augmentation over A .

In the sequel, writing $\mathfrak{q} \wedge \mathfrak{q}'$ will implicitly mean that \mathfrak{q} and \mathfrak{q}' are causally compatible. Again polarities do not matter here. Note also that

Lemma 4.1. *Let $\sigma : S \rightarrow A$, $\tau : T \rightarrow A^\perp$ be two concurrent strategies, then for every causally compatible $\mathfrak{q} \in \text{Rig}(\sigma)$, $\mathfrak{q}' \in \text{Rig}(\tau)$, there is a secured bijection $\varphi \in \mathcal{B}_{\sigma, \tau}^{\text{sec}}$ such that*

$$\sigma \circ \pi_1(\varphi) = (\mathfrak{q} \wedge \mathfrak{q}') = \tau \circ \pi_1(\varphi)$$

as partial order.

Proof. As an augmentation of $\text{Rig}(\sigma)$, \mathfrak{q} is of the form \mathfrak{q}_{x_S} for some $x_S \in \mathcal{C}(S)$; similarly $\mathfrak{q}' = \mathfrak{q}_{x_T}$ for some $x_T \in \mathcal{C}(T)$. It is then immediate to see that $\varphi : x_S \simeq x_T$ defines a secured bijection, by definition of Rig and \wedge . Note that φ is not unique as x_S and x_T might not be. \square

Generalising the above definition to $\mathfrak{q} \in \text{Aug}(A^\perp \parallel B)$ and $\mathfrak{q}' \in \text{Aug}(B^\perp \parallel C)$, we say that they are *causally compatible* if $|\mathfrak{q}| = x_A \parallel x_B$, $|\mathfrak{q}'| = x_B \parallel x_C$, and $(\mathfrak{q} \parallel x_C)$, $(x_A \parallel \mathfrak{q}')$ are causally compatible – where x_A, x_C are regarded as $(x_A, \leq_A) \in \text{Aug}(A)$, $(x_C, \leq_C) \in \text{Aug}(C)$. Their *interaction* is the augmentation over $A \parallel B \parallel C$ defined by

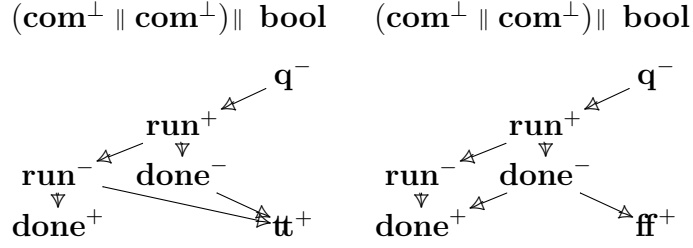
$$\mathfrak{q}' \circledast \mathfrak{q} = (\mathfrak{q} \parallel x_C) \wedge (x_A \parallel \mathfrak{q}').$$

It has configurations of the form $y_A \parallel y_B \parallel y_C$ where $y_A \parallel y_B \in \mathcal{C}(\mathfrak{q})$ and $y_B \parallel y_C \in \mathcal{C}(\mathfrak{q}')$.

There is a slight abuse of notation in the above: we make implicit the renaming (to a ternary parallel composition) that ensures that $\mathfrak{q} \parallel x_C$ and $x_A \parallel \mathfrak{q}'$ both have events in $A \parallel B \parallel C$. We keep this renaming implicit (as is often done in game semantics) because having it explicit overloads notations with no gain in clarity. Moreover, this renaming disappears in the hiding step.

As in CG, augmentations can be projected onto some components of their game.

Definition 4.7. Let $\mathfrak{q} = (|\mathfrak{q}|, \leq_{\mathfrak{q}}) \in \text{Aug}(A)$ and $V \subseteq A$. The *projection* of \mathfrak{q} on V is $\mathfrak{q}_{\downarrow V} = (|\mathfrak{q}| \cap V, \leq_{\mathfrak{q}_{\downarrow V}})$, where $a_1 \leq_{\mathfrak{q}_{\downarrow V}} a_2$ iff $a_1 \leq_{\mathfrak{q}} a_2$. This defines an augmentation over $(A \downarrow V)$.

Figure 4.3: The two maximal augmentations of $[\![\text{strict}]\!]$

In the case of the interaction we get the *composition* of augmentations q, q' as

$$q' \odot q = (q' \otimes q)_{\downarrow(A \parallel C)} \in \text{Aug}(A^\perp \parallel C)$$

Using that we can define the composition of strategies.

Definition 4.8. Let $\sigma : A \xrightarrow{\text{Strat}} B$ and $\tau : B \xrightarrow{\text{Strat}} C$ be two rigid strategies. Their *composition* is

$$\tau \odot \sigma = \{q' \odot q \mid q' \in \tau \ \& \ q \in \sigma \text{ causally compatible}\}$$

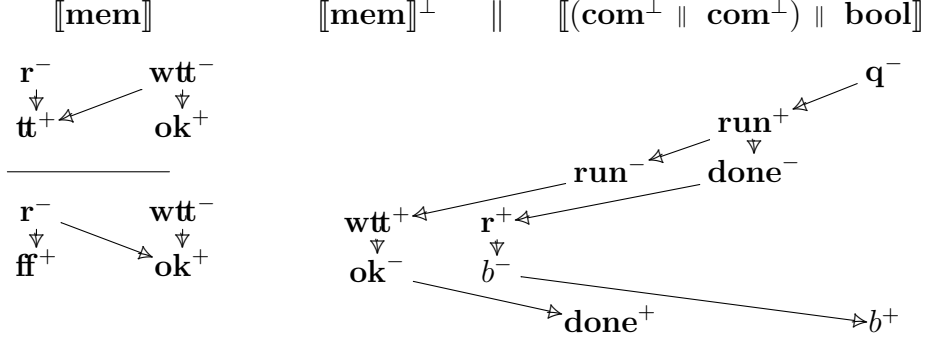
Then, $\tau \odot \sigma : A^\perp \parallel C$ is a rigid strategy.

Example 4.4. For a non-trivial example of composition, we give in figure 4.3 the two maximal (up to rigid inclusion) augmentations of a strategy corresponding to the interpretation of the following term of (affine) IPA:

$$\text{strict} = (\lambda f. \text{newref } r \text{ in } (f(r := \mathbf{tt})); !r) : (\text{com} \rightarrow \text{com}) \rightarrow \text{bool}$$

This term implements a test of strictness: it set a flag r to false and runs the command f on the assignment $r := \mathbf{tt}$, hence returning true only if f calls its argument before returning. This is reflected in the left maximal augmentation by the event \mathbf{tt}^+ being causally dependent from \mathbf{run}^- and \mathbf{done}^- .

The strategy interpreting **strict** is the result of the composition of the strategy with maximal augmentation at the right hand side of figure 4.4 (interpreting the open term $\lambda f. f(r := \mathbf{tt}); !r$ for r a memory location) and the strategy $\text{cell} : [\![\text{mem}]\!]$ that interprets a memory cell, with maximal augmentations at the left hand side of figure 4.4 (the reader may notice that this is actually the rigid image of the strategy cell presented on figure 1.5). Performing composition as in definition 4.8 produces the two maximal augmentations of Figure 4.3.

Figure 4.4: The rigid strategies cell and $\llbracket \lambda f. f (r := \text{tt}); !r \rrbracket$

It is a good exercise to check that the composition as defined in 4.8 preserves courtesy on augmentations and receptivity on rigid strategies. We refrain however from giving the details here as we will show in the next section that composition of rigid strategies is the mirror of composition of strategies in CG via Rig . Instead, we state the following useful lemma:

Lemma 4.2. *For every $\mathfrak{q} \in \tau \odot \sigma$, there is a minimal witness (for prefix inclusion) $\text{wit}(\mathfrak{q}) \in \text{Aug}(\tau \otimes \sigma)$ s.t. $\text{wit}_\sigma(\mathfrak{q}) \downarrow_{A \parallel C} = \mathfrak{q}$.*

Proof. This is direct by definition of $\tau \odot \sigma$. However note that, although minimal, this witness might not be unique. \square

Tensor product As for composition, the tensor product of two rigid strategies is first defined at the level of augmentations then lifted up component-wise. For $\mathfrak{q} \in \text{Aug}(A^\perp \parallel C)$, $\mathfrak{q}' \in \text{Aug}(B^\perp \parallel D)$ we set

$$\mathfrak{q} \otimes \mathfrak{q}' = \gamma * (\mathfrak{q} \parallel \mathfrak{q}') \in \text{Aug}(A^\perp \parallel B^\perp \parallel C \parallel D)$$

where $\gamma : (A \parallel C) \parallel (B^\perp \parallel D) \cong (A^\perp \parallel B^\perp) \parallel (C \parallel D)$ is the same game isomorphism as in section 1.2 but $*$ is the *global renaming* operator defined by

Definition 4.9. Let $f : A \cong A'$ be a game isomorphism and let $\mathfrak{q} \in \text{Aug}(A)$, the *global renaming* of \mathfrak{q} following f is the partial order

$$f * \mathfrak{q} = (f(|\mathfrak{q}|), \leq_{f*\mathfrak{q}})$$

where $f(e) \leq_{f*\mathfrak{q}} f(e')$ iff $e \leq_{\mathfrak{q}} e'$. We have $f * \mathfrak{q} \in \text{Aug}(A')$.

The last statement is easy to see as global renaming leaves the structure of \mathfrak{q} unchanged, in particular

$$\mathcal{C}(f * \mathfrak{q}) = \{f x \mid x \in \mathcal{C}(\mathfrak{q})\}$$

The above remark also implies that renaming preserves \preceq and respect courtesy as well. From there one define, the tensor product of two rigid strategies, $\sigma : A \xrightarrow{\text{Strat}} C, \tau : B \xrightarrow{\text{Strat}} D$ to be

$$\sigma \otimes \tau = \{\mathfrak{q}_\sigma \otimes \mathfrak{q}_\tau \mid \mathfrak{q}_\sigma \in \sigma, \mathfrak{q}_\tau \in \tau\} : (A \otimes B) \xrightarrow{\text{Strat}} (C \otimes D)$$

It is immediate to see that this preserves receptivity.

Pre-order enrichment The partial order \preceq on augmentations can be lifted to rigid strategies component-wise:

Definition 4.10. Let $\sigma, \sigma' : A$ be two rigid strategies, then $\sigma \preceq \sigma'$ iff for every $\mathfrak{q} \in \sigma$ there exists $\mathfrak{q}' \in \sigma'$ such that $\mathfrak{q} \preceq \mathfrak{q}'$.

W.l.o.g., one can assume that $|\mathfrak{q}| = |\mathfrak{q}'|$ (by prefix closure).

From the above, it is immediate to conclude that \otimes preserves \preceq on rigid strategies. The fact that \otimes and \odot preserve \preceq on rigid strategies as well then is a consequence of the following lemma:

Lemma 4.3. Let $\mathfrak{q}_1 \preceq \mathfrak{q}'_1 \in \text{Aug}(A^\perp \parallel B)$ and $\mathfrak{q}_2 \preceq \mathfrak{q}'_2 \in \text{Aug}(B^\perp \parallel C)$ such that $|\mathfrak{q}_1| = |\mathfrak{q}'_1|$, $|\mathfrak{q}_2| = |\mathfrak{q}'_2|$, and $\mathfrak{q}_1, \mathfrak{q}_2$ are causally compatible, then \mathfrak{q}'_1 and \mathfrak{q}'_2 are causally compatible and $\mathfrak{q}_2 \odot \mathfrak{q}_1 \preceq \mathfrak{q}'_2 \odot \mathfrak{q}'_1$.

Proof. By assumption $|\mathfrak{q}_1| = |\mathfrak{q}'_1|$, $|\mathfrak{q}_2| = |\mathfrak{q}'_2|$ and, by compatibility, $|\mathfrak{q}_1| = |\mathfrak{q}_2|$, so $|\mathfrak{q}'_1| = |\mathfrak{q}'_2|$. Moreover, the partial order of \mathfrak{q}'_1 and \mathfrak{q}'_2 are respectively included in the one of \mathfrak{q}_1 and \mathfrak{q}_2 , so the transitive and reflexive closure of their union must be included in the one of $\preceq_{\mathfrak{q}_2 \otimes \mathfrak{q}_1}$ hence define a partial order. The inequalities on interaction and composition follow. \square

Compact closure In the next section we will see that \otimes defines a monoidal structure on rigid strategies. Its structural morphisms correspond to the rigid image of the structural morphisms in CG. Similarly, taking the rigid image of η_A and ν_A for unit and co-unit we will conclude that

Theorem 4.1. Games and rigid strategies, together with \odot , $(1, \otimes)$ and $(_)\perp$ form a compact closed category Strat. It is enriched over \preceq

4.1.2 From CG to Strat

In this section we show that Strat is well-defined as a compact closed category by relating it to CG. More precisely, we complete the map $\text{Rig} : \text{CG} \rightarrow \text{Strat}$ by a backward map $\text{top} : \text{Strat} \rightarrow \text{CG}$

$$\begin{array}{ccc} & \text{Rig} & \\ & \curvearrowright & \\ \text{CG} & & \text{Strat} \\ & \curvearrowleft & \\ & \text{top} & \end{array}$$

such that $\text{Rig} \circ \text{top} = \text{id}_{\text{Strat}}$. We show that Rig and $\text{Rig} \circ \text{top}$ are functorial and that $\text{top}(\text{Rig}(-))$ preserves (up to isomorphism) structural strategies of CG such as copycat . From there we conclude that Strat inherits all the structural properties of CG. This development is not required in the sequel, it is there for completeness.

Primes As intuited in the introduction, rigid strategies correspond to concurrent strategies that are non-deterministically idempotent, that are strategies in which every events is characterised by its corresponding move in the game and its causal history. We make this statement precise by reconstructing a concurrent strategy from a rigid strategy using its *prime* augmentations, that are augmentations with a unique maximal event.

Definition 4.11. Let $\sigma : A$ be a rigid strategy, we define $\text{Pr}(\sigma)$ the event structure with

- Events: prime augmentations of σ , written \mathfrak{q}_a , with a the maximal event of the prime \mathfrak{q} ;
- Causality: $\mathfrak{q}_a \leq \mathfrak{q}_b$ iff $\mathfrak{q}_a \hookrightarrow \mathfrak{q}_b$;
- Consistency: $X \in \text{Con}_{\text{Pr}(\sigma)}$ iff $\exists \mathfrak{p} \in \sigma$ such that for every $\mathfrak{q} \in X$, $\mathfrak{q} \hookrightarrow \mathfrak{p}$.

The mapping $f : \mathfrak{q}_a \mapsto a$ defines a concurrent strategy $\text{top}(\sigma) : \text{Pr}(\sigma) \rightarrow A$.

In the above, consistency ensures that f is locally injective as it implies in particular that any two consistent primes with the same top events must agree on their causal history – in fact two primes are consistent if they agree on the causal history of *all* of their common events. It routine to check that $\text{Pr}(\sigma)$ is a well-defined event structure.

For courtesy and receptivity of $\text{top}(\sigma)$, this is a consequence of σ being courteous and receptive as a rigid strategy. Indeed, keeping in mind that $(x, \leq_{\text{Pr}(\sigma)})$ and \mathfrak{q}_x are order isomorphic (via σ), courtesy and receptivity are inherited from σ following the lemma below:

Lemma 4.4. *Let $\sigma : A$ be a rigid strategy, then the mappings*

$$\begin{array}{ccc} \text{Rig} : & (\mathcal{C}(\text{Pr}|\sigma|), \subseteq) & \rightarrow & (\sigma, \hookrightarrow) \\ & x & \mapsto & \mathfrak{q}_x \end{array}$$

and

$$\begin{array}{ccc} \text{Pr} : & (\sigma, \hookrightarrow) & \rightarrow & (\mathcal{C}(\text{Pr}|\sigma|), \subseteq) \\ & \mathfrak{q} & \mapsto & \{\mathfrak{q}_a = ([a]_{\mathfrak{q}}, \leq_{\mathfrak{q}}) \mid a \in \mathfrak{q}\} \end{array}$$

are order-isomorphic and they preserve the internal partial order. The reuse of $\text{Pr}(-)$, and $\text{Rig}(-)$ shall not induce confusion as it is clear in context on which object they apply.

Proof. First check that Rig is well defined and that \mathfrak{q}_x is an augmentation of σ . By definition $|\mathfrak{q}_x| = (\cup_{\mathfrak{q}_a \in x} a)$ so, by down-closure of x , it is equal to $(\cup_x |\mathfrak{q}_a|)$. Moreover \mathfrak{q}_x and $\cup_x \mathfrak{q}_a$ have the same partial order as $a \leq_{\mathfrak{q}_x} a'$ iff $\mathfrak{q}_a \leq_{\text{Pr}(\sigma)} \mathfrak{q}_{a'}$ iff $a \leq_{\mathfrak{q}_{a'}} a'$, with $\mathfrak{q}_a, \mathfrak{q}_{a'}$ the inverse image of a and a' via f in x . By consistency of x , $\cup_x \mathfrak{q}_a \in \sigma$, so $\mathfrak{q}_x \in \sigma$.

That $\text{Pr}(\mathfrak{q})$ is well-defined is a direct check and $\text{Pr}(-)$ and $\text{Rig}(-)$ are obviously inverse of each other. Moreover it is clear that they preserve order as well as the internal partial order. \square

By definition of $\text{Rig}(-)$, the above lemma also induces that

Corollary 4.1. *Let $\sigma : A$ be a rigid strategy, then $\text{Rig}(\text{top}(\sigma)) = \sigma$.*

The converse is not true, even up to isomorphism, the left concurrent strategy on figure 4.1, call it σ , is different from $\text{top}(\text{Rig}(\sigma))$. However, the strategy on the right, call it σ' is such that $\text{top}(\text{Rig}(\sigma')) \simeq \sigma'$. More generally,

Lemma 4.5. *Let $\sigma : S \rightarrow A$ be a non-deterministically idempotent concurrent strategy then $\text{top}(\text{Rig}(\sigma)) \simeq \sigma$.*

Proof. Given a concurrent strategy $\sigma : S \rightarrow A$, that the mapping

$$\begin{array}{ccc} \varphi : & S & \rightarrow & \text{Pr}(\text{Rig}(\sigma)) \\ & s & \mapsto & \mathfrak{q}_{[s]} \end{array}$$

is a morphism of event structures such that $\sigma = \text{top}(\text{Rig}(\sigma)) \circ \varphi$. By definition, it defines an isomorphism if σ is non-deterministically idempotent. \square

Rig properties. We now show that Rig preserves \simeq , \odot and \otimes from CG.

Lemma 4.6. *Let $\sigma, \tau : A \twoheadrightarrow B$, be two concurrent strategies such that $\sigma \simeq \tau$, then $\text{Rig}(\sigma) = \text{Rig}(\tau)$.*

Proof. Immediate by definition of Rig and \simeq . \square

Lemma 4.7. *Let $\sigma : S \rightarrow A^\perp \parallel B$, $\tau : T \rightarrow B^\perp \parallel C$ be two concurrent strategies, then $\text{Rig}(\tau \odot \sigma) = \text{Rig}(\tau) \odot \text{Rig}(\sigma)$.*

Proof. Let $\mathfrak{q}_{x_T} \odot \mathfrak{q}_{x_S} \in \text{Rig}(\tau) \odot \text{Rig}(\sigma)$, then by definition, \mathfrak{q}_{x_T} and \mathfrak{q}_{x_S} are causally compatible and by lemma 4.1 $x_T \otimes x_S$ is a configuration of $\mathcal{C}(T \otimes S)$ such that $\mathfrak{q}_{x_T \otimes x_S} = \mathfrak{q}_{x_T} \otimes \mathfrak{q}_{x_S}$ so $\mathfrak{q}_{x_T \odot x_S} = \mathfrak{q}_{x_T} \odot \mathfrak{q}_{x_S}$. Conversely, for $\mathfrak{q}_{x_T \odot x_S} \in \text{Rig}(\tau \odot \sigma)$, \mathfrak{q}_{x_T} and \mathfrak{q}_{x_S} are causally compatible and again $\mathfrak{q}_{x_T \odot x_S} = \mathfrak{q}_{x_T} \odot \mathfrak{q}_{x_S}$. So $\text{Rig}(\tau \odot \sigma) = \text{Rig}(\tau) \odot \text{Rig}(\sigma)$. \square

Lemma 4.8. *Let $\sigma : A \twoheadrightarrow C$, $\tau : B \twoheadrightarrow D$ be two concurrent strategies, then $\text{Rig}(\sigma \otimes \tau) = \text{Rig}(\sigma) \otimes \text{Rig}(\tau)$.*

Proof. This is just a matter of checking that for every $x_S \otimes x_T \in \mathcal{C}(S \otimes T)$, $\mathfrak{q}_{x_S \otimes x_T} = \mathfrak{q}_{x_S} \otimes \mathfrak{q}_{x_T}$, which is direct following the definitions above. \square

top properties As for Rig, top preserves \otimes .

Lemma 4.9. *Let $\sigma : A \xrightarrow{\text{Strat}} C$, $\tau : B \xrightarrow{\text{Strat}} D$ be two rigid strategies, then $\text{top}(\sigma \otimes \tau) \simeq \text{top}(\sigma) \otimes \text{top}(\tau)$.*

Proof. Using lemma 4.4 there is an obvious isomorphism between the two strategies, derived from the order-isomorphism:

$$\begin{aligned} \mathcal{C}(\text{Pr}(\sigma \otimes \tau)) &\rightarrow \mathcal{C}(\text{Pr}(\sigma)) \parallel \mathcal{C}(\text{Pr}(\tau)) \\ \text{Pr}(\mathfrak{q}_\sigma \otimes \mathfrak{q}_\tau) &\mapsto \text{Pr}(\mathfrak{q}_\sigma) \parallel \text{Pr}(\mathfrak{q}_\tau) \end{aligned}$$

\square

However, top on its own does not preserve the composition of rigid strategies. Yet:

Lemma 4.10. *Let $\sigma : A \xrightarrow{\text{Strat}} B$, $\tau : B \xrightarrow{\text{Strat}} C$ be two rigid strategies, then $\text{Rig}(\text{top}(\tau \odot \sigma)) = \text{Rig}(\text{top}(\tau) \odot \text{top}(\sigma))$.*

Proof. By corollary 4.1 this is equivalent to show that $\tau \odot \sigma = \text{Rig}(\text{top}(\tau) \odot \text{top}(\sigma))$.

Let $\mathfrak{q} \in \tau \odot \sigma$, then by lemma 4.2 it has a minimal witness $\mathfrak{q}_\tau \otimes \mathfrak{q}_\sigma \in \tau \otimes \sigma$. But, by lemma 4.4, $\mathfrak{q}_\sigma, \mathfrak{q}_\tau$ are causally compatible iff $\text{Pr}(\mathfrak{q}_\sigma) \in \text{top}(\sigma)$

and $\text{Pr}(q_\tau) \in \text{top}(\tau)$ define a secured bijection. Hence $\text{Pr}(q_\tau) \odot \text{Pr}(q_\sigma) \in \text{top}(\tau) \odot \text{top}(\sigma)$ and is such that $\text{Rig}(\text{Pr}(q_\tau) \odot \text{Pr}(q_\sigma)) = \mathfrak{q}$. So $\tau \odot \sigma \subseteq \text{Rig}(\text{top}(\tau) \odot \text{top}(\sigma))$.

Reciprocally, let $x_{\text{top}(\tau)} \odot x_{\text{top}(\sigma)} \in \text{top}(\tau) \odot \text{top}(\sigma)$, then, by lemma 4.4 again, there is $q_\sigma \in \sigma$ and $q_\tau \in \tau$ such that $\text{Pr}(q_\sigma) = x_{\text{top}(\sigma)}$, $\text{Pr}(q_\tau) = x_{\text{top}(\tau)}$ and q_σ, q_τ are causally compatible. So $q_\tau \odot q_\sigma \in \tau \odot \sigma$ and is such that $q_\tau \odot q_\sigma = \text{Rig}(x_{\text{top}(\tau)} \odot x_{\text{top}(\sigma)})$. Hence $\tau \odot \sigma \supseteq \text{Rig}(\text{top}(\tau) \odot \text{top}(\sigma))$. \square

Structure inheritance. The five previous lemmas and corollary 4.1 induce that, by inheritance from CG, the composition of rigid strategies is associative and that the tensor product is functorial in Strat. Let us do the case for associativity – the rest works similarly. Let $\sigma : A \xrightarrow{\text{Strat}} B$, $\tau : B \xrightarrow{\text{Strat}} C$, $\rho : C \xrightarrow{\text{Strat}} D$ be rigid strategies, then

$$\begin{aligned}
(\rho \odot \tau) \odot \sigma &= \text{Rig}(\text{top}((\rho \odot \tau) \odot \sigma)) \\
&= \text{Rig}((\text{top}(\rho) \odot \text{top}(\tau)) \odot \text{top}(\sigma)) \\
&= \text{Rig}(\text{top}(\rho \odot \tau)) \odot \text{Rig}(\text{top}(\sigma)) \\
&= \text{Rig}(\text{top}(\rho) \odot \text{top}(\tau)) \odot \text{Rig}(\text{top}(\sigma)) \\
&= \text{Rig}((\text{top}(\rho) \odot \text{top}(\tau)) \odot \text{top}(\sigma)) \\
&= \text{Rig}(\text{top}(\rho) \odot (\text{top}(\tau) \odot \text{top}(\sigma))) \\
&= \text{Rig}(\text{top}(\rho)) \odot \text{Rig}((\text{top}(\tau) \odot \text{top}(\sigma))) \\
&= \text{Rig}(\text{top}(\rho)) \odot \text{Rig}(\text{top}(\tau \odot \sigma)) \\
&= \text{Rig}(\text{top}(\rho \odot (\tau \odot \sigma))) \\
&= \rho \odot (\tau \odot \sigma)
\end{aligned}$$

For structural morphisms, one can note that concurrent copycat strategies are injective so they are non-deterministically idempotent: given a game A , $\mathfrak{c}_A(a) = \mathfrak{c}_A(a')$ iff $a = a'$. By lemma 4.5, this implies in particular that for every $f : A \cong A'$ (including $f = \text{id}_A$), $\text{top}(\text{Rig}(\mathfrak{c}_f)) \simeq \mathfrak{c}_f$. From there one can check that the rigid image of every structural morphism in CG is a structural morphism in Strat. Let us do the case for \mathfrak{c}_A – others are similar. Let $\rho : A \xrightarrow{\text{Strat}} B$ a rigid strategy,

$$\begin{aligned}
\rho \odot \text{Rig}(\mathfrak{c}_A) &= \text{Rig}(\text{top}(\rho \odot \text{Rig}(\mathfrak{c}_A))) \\
&= \text{Rig}((\text{top}(\rho) \odot \text{top}(\text{Rig}(\mathfrak{c}_A)))) \\
&= \text{Rig}(\text{top}(\rho) \odot \mathfrak{c}_A) \\
&= \text{Rig}(\text{top}(\rho)) \\
&= \rho
\end{aligned}$$

Note that $\text{Rig}(\mathfrak{c}_f)$ can be rephrased in terms of rigid copycat strategies using *local left renaming*:

Definition 4.12. Let $f : A \cong A'$ be a game isomorphism and let $\mathfrak{q} \in \text{Aug}A^\perp \parallel B$, then the *local left renaming* of \mathfrak{q} via f is the partial order

$$(f \cdot \mathfrak{q}) = (f \parallel B) * \mathfrak{q}$$

Obviously $f \cdot \mathfrak{q} \in \text{Aug}((A')^\perp \parallel B)$.

It is a simple check to see $\text{Rig}(\mathfrak{c}_f) = f \cdot (\text{Rig}(\mathfrak{c}_A))$.

4.2 Elementary games and strategies

Rideau and Winskel's concurrent games are suited to capture both concurrency and non-determinism/conflicts. In the previous section we have recalled how one can simplify the setting and restrict the kind of non-determinism allowed in strategies. We now present an other (and more drastic) simplification that consists in restricting concurrent games to only *elementary games*, that are, games with *no conflicts*, and strategies upon them to elementary strategies, that are strategies with no conflicts either. This leads to a purely deterministic (but still concurrent) model, named Det, which has direct connections with both CG and Strat.

4.2.1 The compact closed structure of Det

Removing conflicts, event structures become partial orders, sometimes called *elementary event structures* (ees). Similarly to augmentations, an ees A is described as $A = (|A|, \leq_A)$ with $|A|$ its set of *events* and \leq_A its causal (partial) order, that enjoy the finite history property of event structure.

All the operations and notations defined in 1.1 can be restricted to ees. In particular, ees and their morphisms form a monoidal subcategory of \mathcal{E} .

Elementary games are defined as ees with polarities.

Definition 4.13. An *elementary game* is an ees $A = (|A|, \leq_A, \text{pol}_A)$ with $(|A|, \leq_A)$ and a *polarity function*

$$\text{pol}_A : |A| \rightarrow \{+, -\}$$

Parallel composition and the dual operation $(-)^{\perp}$ are obviously stable on elementary games.

Elementary strategies Elementary games are games without conflict, they form a subset of concurrent games that are closed under dual and tensor product. Informally, elementary strategies are concurrent strategies on elementary games that do not have internal conflict either. In CG such strategies correspond to morphisms $\sigma : S \rightarrow A$ where S and A are both ees. In Strat this matches with rigid strategies that have a unique maximal augmentation for the prefix relation. We will see that in both cases, this can be simplified by the data of a single augmentation.

Definition 4.14. A *elementary strategy* on an elementary game A is an ees $\sigma = (|\sigma|, \leq_\sigma) \in \text{Aug}(A)$ that verifies the following two additional conditions:

- *Receptivity.* If $x \in \mathcal{C}(\sigma)$ and $x \xrightarrow{a^-} \text{C}$ in A (meaning $x \cup \{a\} \in \mathcal{C}(A)$), then $a \in |\sigma|$.
- *Courtesy.* If $a_1 \rightarrow_\sigma a_2$ and $(\text{pol}_A(a_1) = + \text{ or } \text{pol}_A(a_2) = -)$, then $a_1 \rightarrow_A a_2$ as well.

We write $\sigma : A$ to mean that σ is a strategy on the elementary game A .

Dually to the remark above, one can regard $\sigma : A$ as a concurrent strategy in the usual sense through the identity-on-events map of event structures $\text{id} : \sigma \rightarrow A$. The conditions on elementary strategies are almost exactly the same as the standard ones, except receptivity which is slightly ‘‘optimized’’. For completeness, the lemma below relates the receptivity condition above to the usual one.

Lemma 4.11. *Take $\sigma \in \text{Aug}(A)$ satisfying courtesy. Then, it is receptive iff for all $x \in \mathcal{C}(A)$ such that $x \xrightarrow{a^-} \text{C}$ in A , $x \cup \{a\} \in \mathcal{C}(\sigma)$.*

Proof. Right to left is obvious; Left to right requires courtesy: If $x \xrightarrow{a^-} \text{C}$, then by receptivity we know that $a \in |\sigma|$. If $a' \rightarrow_\sigma a$ then, by courtesy, we have $a' \rightarrow_A a$ as well. Since $x \cup \{a\} \in \mathcal{C}(A)$, then we know that for all $a' \rightarrow_A a$, we have $a' \in x$. Therefore, for all $a' \rightarrow_\sigma a$, we have $a' \in x$. It follows that $x \cup \{a\} \in \mathcal{C}(\sigma)$ as well. \square

Because of the absence of event renaming in elementary strategies, it worth noting that isomorphism of elementary strategies in CG yields equality:

Lemma 4.12. *If $\sigma, \sigma' : A$ are two elementary strategies such that $\sigma \simeq \sigma'$ then $\sigma = \sigma'$.*

Proof. By definition there is an isomorphism of ees $\varphi : \sigma \cong \sigma'$ such that $\text{id}_\sigma = \text{id}_{\sigma'} \circ \varphi$. Hence $\varphi = \text{id}_\sigma$. \square

Composition. As previously, an elementary strategy defines a morphism from an elementary game A to elementary game B if it plays on $A^\perp \parallel B$. We write $\sigma : A \xrightarrow{\text{Det}} B$.

Being a particular case of regular plain strategies, elementary strategies can be composed in CG, and it is clear enough that the result has no conflicts. However, composition as defined in section 1.2.3 performs a renaming on the events of the game. In what follows we give an alternative definition for the composition of elementary strategies and show that it is isomorphic to the usual composition in CG. Using the wording of augmentations, we first introduce the notion of *secured event*.

Definition 4.15. Let $\sigma : A$ and $\tau : A^\perp$ be two elementary strategies, an event $a \in |\sigma| \cap |\tau|$ is *secured* if there exist two prefixes $\mathfrak{q} \hookrightarrow \sigma$, $\mathfrak{p} \hookrightarrow \tau$ such that $a \in |\mathfrak{q}| \cap |\mathfrak{p}|$ and $\mathfrak{q}, \mathfrak{p}$ are causally compatible.

Note that if a is secured, then there exists a unique *minimal* pair of such causally compatible prefixes, obtained as the intersection of all possible pairs. In this case $\mathfrak{q} \wedge \mathfrak{p}$ is prime with maximal event a , we write it \mathfrak{q}_a . Following lemma 4.1, there is a one to one correspondence between secured events and prime secured bijection in the interaction of σ and τ viewed as concurrent strategies. More generally,

Proposition 4.1. Let $\sigma : A$ and $\tau : A^\perp$ be two elementary strategies, their interaction $\sigma \wedge \tau : A$ in CG is isomorphic to the augmentation over A defined by

- Events: secured events in $|\sigma| \cap |\tau|$;
- Causality: $(\leq_\sigma \cup \leq_\tau)^*$ restricted to secured events;

and viewed as a concurrent strategy via the identity-on-event morphism.

Proof. From the remark above, the projections $\Pi_1 = \Pi_2$ define a bijection between $|\sigma \wedge \tau|$ and $\{a \in |\sigma| \otimes |\tau| \mid a \text{ is secured}\}$. Moreover, $\sigma \wedge \tau$ and $\text{id}_A \circ \Pi_1$ are equal by definition. For causality, one follows lemma 1.5 stating that $[a, a]_\varphi \leq_{\sigma \wedge \tau} [a', a']_{\varphi'}$ iff $a \leq_\varphi a'$ which is equivalent to $(\leq_\sigma \cup \leq_\tau)^*$ by lemma 4.1. \square

This alternative version of $\sigma \wedge \tau$ defines an elementary strategy, it is the *elementary interaction* of σ and τ , also written $\sigma \wedge \tau$. This extends to

Definition 4.16. Let $\sigma : A \xrightarrow{\text{Det}} B$ and $\tau : B \xrightarrow{\text{Det}} C$ be two elementary strategies, their *interaction* $\tau \otimes \sigma$ is the augmentation:

$$\tau \otimes \sigma = (\sigma \parallel C) \wedge (A \parallel \tau) \in \text{Aug}(A \parallel B \parallel C)$$

This operation is well-defined as σ, C, A, τ are all ees and parallel composition is defined on ees. Yet, there is again a slight abuse of notation (an implicit renaming to a ternary parallel composition) ensuring that $\sigma \parallel C$ and $A \parallel \tau$ both have events a subset of those of $A \parallel B \parallel C$. This renaming disappears with hiding:

Definition 4.17. Let $\sigma : A \xrightarrow{\text{Det}} B$ and $\tau : B \xrightarrow{\text{Det}} C$ be two elementary strategies. Their *composition* is the augmentation:

$$\tau \odot \sigma = (\tau \otimes \sigma)_{\downarrow(A \parallel C)}$$

which is an elementary strategy by definitions 1.13 and 4.14.

From definition 4.17, $\tau \odot \sigma$ has events the secured events that are visible in $A^\perp \parallel C$, partially ordered by restriction of the graph $(\leq_\sigma \cup \leq_\tau)^*$. In particular, every configuration $x \in \mathcal{C}(\tau \otimes \sigma)$ is of the form

$$x_A \parallel x_B \parallel x_C$$

such that $x_A \parallel x_B \in \mathcal{C}(\sigma)$, $x_B \parallel x_C \in \mathcal{C}(\tau)$ and $(\leq_\sigma \cup \leq_\tau)^* \cap x^2$ defines a partial order. Finally, for every $x \in \mathcal{C}(\tau \odot \sigma)$ there is a unique *minimal witness* for x :

$$[x]_{\tau \otimes \sigma} \in \mathcal{C}(\tau \otimes \sigma)$$

Categorical structure Composition in Det inherits its categorical properties from composition in CG. Writing \odot_{CG} for the usual composition in CG, let us first relate the two:

Lemma 4.13. Let $\sigma : A \xrightarrow{\text{Det}} B$ and $\tau : B \xrightarrow{\text{Det}} C$ be two elementary strategies, then $\tau \odot \sigma \simeq \tau \odot_{\text{CG}} \sigma$ in CG.

Proof. This is clear from proposition 4.1, as projection preserve isomorphisms. \square

Associativity follows as an immediate corollary:

Corollary 4.2. Let $\sigma : A \xrightarrow{\text{Det}} B$, $\tau : B \xrightarrow{\text{Det}} C$, $\rho : C \xrightarrow{\text{Det}} D$ be three elementary strategies, then

$$\rho \odot (\tau \odot \sigma) = (\rho \odot \tau) \odot \sigma$$

Proof. By lemma 4.13 and associativity of \odot_{CG}

$$\rho \odot (\tau \odot \sigma) \simeq (\rho \odot_{\text{CG}} \tau) \odot_{\text{CG}} \sigma \simeq \rho \odot_{\text{CG}} (\tau \odot_{\text{CG}} \sigma) \simeq \rho \odot (\tau \odot \sigma)$$

By lemma 4.12, this leads to the equality of the outer elementary strategies. \square

There is no change to be made on the definition of copycat strategies as they already define elementary strategies on elementary games – their consistency matching with those of the game they play on. By inheritance from CG and following lemmas 4.13 and 4.12 again, they are neutral for the above composition with respect to equality.

Concurrent copycat strategies on elementary games match the definition of elementary strategies – recall that for every game A , $\mathfrak{c}_A : \mathbb{C}_A \rightarrow A^\perp \parallel A$ is the-identity-on-event map. Using lemmas 4.13 and 4.12 as above, one can see that for every elementary $\sigma : A \xrightarrow{\text{Det}} B$,

$$\sigma \odot \mathfrak{c}_A \simeq \sigma \odot_{\text{CG}} \mathfrak{c}_A \simeq \sigma$$

hence $\sigma \odot \mathfrak{c}_A = \sigma$ and so copycat strategies are also neutrals for the elementary composition.

In the end, we get

Proposition 4.2. *Elementary games and strategies define a category, written Det.*

We conclude this section by showing how Det inherits of the compact closed structure of CG.

Compact closed structure Following CG, the tensor product of two elementary games A and B is still defined by $A \otimes B = A \parallel B$. As in Strat, we use global renaming to define the tensor product of two elementary strategies.

Definition 4.18. Let $\sigma : A \xrightarrow{\text{Det}} C$, $\tau : B \xrightarrow{\text{Det}} D$ be two elementary strategies, their *tensor product* in Det is

$$\sigma \otimes \tau = \gamma * (\sigma \parallel \tau) : (A \otimes B)^\perp \parallel (C \otimes D)$$

with $\gamma : (A \parallel C) \parallel (B^\perp \parallel D) \cong (A^\perp \parallel B^\perp) \parallel (C \parallel D)$.

This leads to an elementary strategy whose configurations are exactly those of the form

$$(x_A \parallel x_B) \parallel (x_C \parallel x_D) \in \mathcal{C}((A \parallel B)^\perp \parallel (C \parallel D))$$

such that $x_A \parallel x_C \in \mathcal{C}(\sigma_1)$ and $x_B \parallel x_D \in \mathcal{C}(\sigma_2)$.

Again, the alternative definition for tensor on elementary strategies is isomorphic to the usual one, denoted \otimes_{CG} in CG:

Lemma 4.14. *Let $\sigma : A \xrightarrow{\text{Det}} C$ and $\tau : B \xrightarrow{\text{Det}} D$ be two elementary strategies, then $\sigma \otimes \tau \simeq \sigma \otimes_{\text{CG}} \tau$ in CG.*

This is a consequence of the more general lemma:

Lemma 4.15. *Let $\sigma : A$ be an elementary strategy and let $f : A \cong A'$ be an isomorphism of ees, then $f * \sigma \simeq f \circ \sigma$ in CG.*

Proof. By definition of global renaming, $f_{|\sigma|}$ defines an isomorphism of event structures $|\sigma| \cong |f * \sigma|$ and clearly $f \circ \text{id}_A = \text{id}_{A'} \circ f_{|\sigma|}$ so their corresponding strategies are isomorphic in CG. \square

Following the same reasoning as for associativity of composition and neutrality of copycats, bifactoriality of \otimes_{CG} and the above lemma implies that the tensor product on elementary strategies is a bifunctor

$$- \otimes - : \text{Det} \times \text{Det} \rightarrow \text{Det}$$

In CG, the structural strategies that turn \otimes into a monoidal structure are defined as left renamings of copycat strategies. These renamings are based on post-compositions of the strategies regarded as and do not define elementary strategies. However they can simply be adapted to elementary strategies by global renaming; we reset:

Definition 4.19. Let $f : A \cong A'$ be an ees isomorphism and let \mathfrak{q} be an augmentation over $A^\perp \parallel B$, the *local left renaming* of \mathfrak{q} via f is the partial order

$$(f \cdot \mathfrak{q}) = (f \parallel B) * \mathfrak{q}$$

Obviously $f \cdot \mathfrak{q} \in \text{Aug}((A')^\perp \parallel B)$.

Following lemma 4.15 again, for σ an elementary strategy, the elementary version of $f \cdot \sigma$ is isomorphic to its usual version in CG and the monoidal structure of \otimes in CG is thus transferred onto Det.

Finally, as for copycat, the unit η_A and co-unit ν_A in CG define elementary strategies on elementary games, so following again the same reasoning, one can conclude that the compact closed structure onto CG is transferred onto Det.

Theorem 4.2. *Elementary games and strategies, together with \odot , $(1, \otimes)$ and $(_)^\perp$ form a compact closed category Det.*

In fact, Det is also order enriched by \preceq defined as previously on partial orders. The congruence of \otimes and \odot with respect to \preceq is direct from the characterisation their configurations (below definitions 4.18 and 4.17). Alternatively, this is a consequence of the connection between Det and Strat described in the following section.

4.2.2 Relation with Strat

As mentioned when introduced, elementary strategies can be viewed through two different prisms:

1. either as a subset of the usual concurrent strategies on elementary games, the restriction then corresponds to strategies being only allowed to have for events a subset of moves of their game, considering that renaming events is unnecessary without non-determinism;
2. or as a simplification of rigid strategies on elementary games as only one augmentation is then needed to fully describe a strategy.

In this section we comment on the second point. The objective is to give a better understanding of the three models but its content will not be required in the sequel.

Fat elementary strategies As mentioned in example 4.2, the prefix closure of an elementary strategy $\sigma : A$ gives rise to a rigid strategy $\text{Rig}(\sigma) : A$, also called the *fat* version of σ , written $\text{Fat}(\sigma)$. By definition, $\text{Fat}(\sigma)$ has a unique maximal augmentation, σ .

Conversely, given an elementary game A and a rigid strategy $\sigma : A$ with a unique maximal augmentation for prefix, its *meager* version, $\text{Meager}(\sigma) = \max(\sigma)$, defines an elementary strategy on A .

Obviously, $\text{Meager} \circ \text{Fat} = \text{id}_{\text{Det}}$, and $\text{Fat} \circ \text{Meager}$ is the identity where it is defined. Thus there is a bijection between elementary strategies and rigid strategies over elementary games with a unique maximal augmentation for the rigid inclusion, we call those last strategies *rigid elementary strategies*.

One can note that, as $\text{Fat}(\sigma)$ is actually equal to $\text{Rig}(\sigma)$, then by lemmas 4.6, 4.7 and 4.8, it is functorial. In particular since $\text{Meager}(\text{Fat}(\sigma)) = \max(\text{Fat}(\sigma)) = \sigma$, this yields an interesting corollary, translated (on the first line) the fact that the interaction between two elementary strategies correspond to the maximal partial order on which they both agree.

Corollary 4.3. *Let $\sigma : A \xrightarrow{\text{Det}} B$, $\tau : B \xrightarrow{\text{Det}} C$ and $\rho : D \xrightarrow{\text{Det}} C$ then these three equalities hold:*

$$\tau \circledast \sigma = \max(\text{Rig}(\tau) \circledast \text{Rig}(\sigma))$$

$$\tau \odot \sigma = \max(\text{Rig}(\tau) \odot \text{Rig}(\sigma))$$

$$\rho \otimes \sigma = \max(\text{Rig}(\rho) \otimes \text{Rig}(\sigma))$$

Meager rigid strategies The fat and meager relation between elementary strategies and rigid elementary strategies can be extended to the whole set of rigid strategies by considering *meager strategies*:

Definition 4.20. Let A be a game, a *meager* strategy over A is a set of courteous and receptive augmentations, $\sigma \subseteq \text{Aug}(A)$, such for every $\mathfrak{q}, \mathfrak{q}' \in \sigma$, \mathfrak{q} and \mathfrak{q}' are incomparable for rigid inclusion.

Meager and Fat (extended to sets of augmentation component-wise) define an isomorphism between the set of rigid strategies, partially order by set-inclusion, and the set of meager strategies, partially ordered by prefix inclusion. Meager strategies are thus useful to compare or represent (fat) rigid strategies. As already shown on various examples, we will mostly use the meager view on rigid strategies when describing them.

However, if a tensor product can be defined on meager strategies by lifting the tensor product on augmentations component-wise, the composition of meager strategies using the elementary composition component-wise on their maximal augmentations is not well-defined: it may be that the interaction of two maximal augmentations results in a non-maximal augmentations. The representation of rigid strategies through their meager form thus should not be misleading: if considering a composition, this will actually refer to the usual composition on rigid strategies, letting,

$$(\tau \odot \sigma) = \text{Meager}(\text{Fat}(\tau) \odot \text{Fat}(\sigma))$$

for $\sigma : A \xrightarrow{\text{Strat}} B$, $\tau : B \xrightarrow{\text{Strat}} C$ two meager strategies.

4.3 Putting annotations back

In this section we show that the two simplified model of CG presented above extend well with annotations on strategies. As Det and Strat are both based on augmentations, and these are closely related to configurations of concurrent strategies, we take the reverse path from the previous sections and first define a notion of annotated augmentations inspired by the configurations of annotated concurrent strategies. We do it in the general case of annotations by an inequational theory \mathbb{T} .

4.3.1 \mathbb{T} -augmentations

Consider $\mathbb{T} = (\Sigma, \mathbb{E})$ an inequational theory. Recall that $\mathbb{T}(\mathcal{V})$, the free \mathbb{T} -algebra over \mathcal{V} , is defined by $(\text{Tm}_{\Sigma}(\mathcal{V}) / \equiv_{\mathbb{E}}, \{-\}_{\mathbb{E}})$ so that substitution is stable

with respect to $\equiv_{\mathbb{E}}$ and $\leq_{\mathbb{E}}$. Also recall that an assignment $\rho : \mathcal{V} \rightarrow \mathbb{T}(\mathcal{V})$ is said to be *idempotent*: if for every $x \in \mathcal{V}$, $\{\rho(v)\}_{\mathbb{E}}[\rho] = \{\rho(v)\}_{\mathbb{E}}$.

We now define \mathbb{T} -augmentations:

Definition 4.21. A \mathbb{T} -*augmentation* over an event structure A is an augmentation $\mathfrak{q} \in \text{Aug}(A)$ together with an *idempotent* function

$$\lambda_{\mathfrak{q}} : |\mathfrak{q}| \rightarrow \mathbb{T}(|\mathfrak{q}|)$$

called the *annotation* of \mathfrak{q} .

We write $\mathbb{T}\text{-Aug}(A)$ for the set of \mathbb{T} -augmentations on A . It is partially ordered by $\mathfrak{q} \preceq \mathfrak{q}'$ iff $\mathfrak{q} \preceq \mathfrak{q}'$ as plain augmentations (def 4.3) and for every $a \in \mathfrak{q}$, $\lambda_{\mathfrak{q}}(a) \leq_{\mathbb{E}} \lambda_{\mathfrak{q}'}(a)$.

Example 4.5. Given a \mathbb{T} -concurrent strategy $(\sigma, \lambda_{\sigma}) : S \rightarrow A$, and a configuration $x \in \mathcal{C}(S)$, its rigid image \mathfrak{q}_x extends to a \mathbb{T} -augmentation by setting:

$$\begin{aligned} \lambda_{\mathfrak{q}_x}(\sigma(s^+)) &= \lambda_{\sigma}(s^+)[\sigma] \\ \lambda_{\mathfrak{q}_x}(\sigma(s^-)) &= \{\sigma(s)\}_{\mathbb{E}} \end{aligned}$$

From now on, \mathfrak{q}_x will stand for $(\mathfrak{q}_x, \lambda_{\mathfrak{q}_x})$, the \mathbb{T} -rigid image of x .

Note that \mathbb{T} -augmentations are more general than what is expected from the rigid image of the configurations in a \mathbb{T} -strategy. In particular, they are defined on both positive and negative moves. This generality is necessary for talking about augmentations that are not defined on a game but only on event structure without polarity as, for example, the interactions of two causally compatible augmentations. For strategies, however, we will only be interested in certain kinds of augmentations that satisfy two sanity properties:

Definition 4.22. Let A be a game and $\mathfrak{q} \in \mathbb{T}\text{-Aug}$, then \mathfrak{q} is

- \mathbb{T} -*receptive* if for every $a^- \in |\mathfrak{q}|$, $\lambda_{\mathfrak{q}}(a) = \{a\}_{\mathbb{E}}$;
- \mathbb{T} -*courteous* if for every $a^+ \in |\mathfrak{q}|$, $\lambda_{\mathfrak{q}}(a) \in \mathbb{T}([a]_{\mathfrak{q}}^-)$;

In the above, \mathbb{T} -courtesy is not surprising as it matches with the definition of annotations on positive moves in \mathbb{T} -concurrent strategies. \mathbb{T} -receptivity on the other side should not cause more surprise as it only makes explicit the fact that annotations on negative moves are open to any values provided by Opponent.

One can check that the \mathbb{T} -rigid image of a \mathbb{T} -concurrent strategy yields \mathbb{T} -receptive and \mathbb{T} -courteous augmentations. In fact following the same completion of positive annotations with negative equivalence classes as presented in example 4.5, it is easy to see that:

Lemma 4.16. *Let A be a game and $\mathfrak{q} \in \text{Aug}(A)$, then there is a one-to-one correspondence between \mathbb{T} -courteous and \mathbb{T} -receptive annotations for \mathfrak{q} and functions of the form $\lambda : a \in |\mathfrak{q}|^+ \rightarrow \mathbb{T}([a]_{\mathfrak{q}}^-)$.*

Composition We now show how constructions on augmentations extend to \mathbb{T} -augmentations. Starting with interaction, we say that two \mathbb{T} -augmentations $\mathfrak{q} \in \mathbb{T}\text{-Aug}(A^\perp \parallel B)$, $\mathfrak{q}' \in \mathbb{T}\text{-Aug}(B^\perp \parallel C)$ are *causally compatible* if their underlying augmentations are.

Definition 4.23. Let $\mathfrak{q} \in \mathbb{T}\text{-Aug}(A^\perp \parallel B)$, $\mathfrak{q}' \in \mathbb{T}\text{-Aug}(B^\perp \parallel C)$ be two causally compatible \mathbb{T} -courteous and \mathbb{T} -receptive augmentations, their *interaction* is $\mathfrak{q}' \otimes \mathfrak{q}$ together with the annotation $\lambda_{\mathfrak{q}' \otimes \mathfrak{q}}$ inductively defined on $\prec_{\mathfrak{q}' \otimes \mathfrak{q}}$ by:

$$\begin{aligned} \lambda_{\mathfrak{q}' \otimes \mathfrak{q}} : \quad & |\mathfrak{q}' \otimes \mathfrak{q}| && \rightarrow && \text{Im}_\sigma(|\mathfrak{q}' \otimes \mathfrak{q}|) \\ e \in (A^\perp \parallel B^+) & \mapsto && \lambda_{\mathfrak{q}}(e)[\lambda_{\mathfrak{q}' \otimes \mathfrak{q}}] \\ e' \in (B^- \parallel C) & \mapsto && \lambda_{\mathfrak{q}'}(e')[\lambda_{\mathfrak{q}' \otimes \mathfrak{q}}] \end{aligned}$$

We have $(\mathfrak{q}' \otimes \mathfrak{q}, \lambda_{\mathfrak{q}' \otimes \mathfrak{q}}) \in \mathbb{T}\text{-Aug}(A \parallel B \parallel C)$

In the above, again, we keep renaming to ternary parallel composition silent. More interestingly, note that if $e \in (A^\perp \parallel C)^-$, then $\lambda_{\mathfrak{q}' \otimes \mathfrak{q}}(e) = \{e\}_{\mathbb{E}}$. This resembles very much the annotation for the interaction of two \mathbb{T} -strategies (see lemma 2.3) but with a lighter writing as the two causally compatible augmentations share the same set of variables. In fact, following lemma 4.1, we have

Lemma 4.17. *Let $\sigma : S \rightarrow A$, $\tau : T \rightarrow A^\perp$ be two concurrent strategies, and let $\mathfrak{q}_{x_S} \in \text{Rig}(\sigma)$, $\mathfrak{q}_{x_T} \in \text{Rig}(\tau)$ such that $x_T \otimes x_S \in \mathcal{C}(T \otimes S)$, then*

$$(\tau \otimes \sigma).(\lambda_{\tau \otimes \sigma})|_{x_T \otimes x_S} = \lambda_{\mathfrak{q}_{x_T} \otimes \mathfrak{q}_{x_S}}$$

Proof. From lemma 1.5 and lemma 4.1, $(\tau \otimes \sigma)(x_T \otimes x_S) = \mathfrak{q}' \otimes \mathfrak{q}$ as partial orders. The above follows by induction on $\prec_{x_T \otimes x_S}$

- if $[s, t]_{x_T \otimes x_S}$ such that $s \in S^+$ then $(\tau \otimes \sigma)([s, t]_{x_T \otimes x_S}) = e \in (A^\perp \parallel B)^+$ and $\lambda_{\mathfrak{q}_{x_T} \otimes \mathfrak{q}_{x_S}}(e) = \lambda_{\mathfrak{q}_{x_S}}(e)[\lambda_{\mathfrak{q}_{x_T} \otimes \mathfrak{q}_{x_S}}] = \lambda_\sigma(s)[\sigma][\lambda_{\mathfrak{q}_{x_T} \otimes \mathfrak{q}_{x_S}}]$ by definition. Then, by induction, this is equal to $\lambda_\sigma(s)[\sigma][(\tau \otimes \sigma).(\lambda_{\tau \otimes \sigma})|_{x_T \otimes x_S}] = \lambda_\sigma(s)[\Pi_1^{-1}][(\lambda_{\tau \otimes \sigma})|_{x_T \otimes x_S}][\tau \otimes \sigma] = (\tau \otimes \sigma).(\lambda_{\tau \otimes \sigma})|_{x_T \otimes x_S}(e)$;
- the same hold with λ_τ if $[s, t]_{x_T \otimes x_S}$ such that $t \in T^+$ as in that case $\mathbb{T} \otimes S([s, t]_{x_T \otimes x_S}) = e' \in (B^\perp \parallel C)^+$;

- otherwise, $\mathbb{T} \otimes S(\llbracket s, t \rrbracket_{x_T \otimes x_S}) = a \in (A^\perp \parallel C)^-$ and by \mathbb{T} -courtesy of $\lambda_{q_{x_S}}$ and $\lambda_{q_{x_T}}$, $\lambda_{q_{x_T} \otimes q_{x_S}}(a) = \{a\}_{\mathbb{E}} = (T \otimes S).(\lambda_{\tau \otimes \sigma}) \upharpoonright_{x_{x_T \otimes x_S}}(a)$.

□

Having define interaction, we can now extend the construction to composition

Definition 4.24. Let $q \in \mathbb{T}\text{-Aug}(A^\perp \parallel B)$, $q' \in \mathbb{T}\text{-Aug}(B^\perp \parallel C)$ be two causally compatible \mathbb{T} -courteous and \mathbb{T} -receptive augmentations, their *composition* is $q' \odot q$ together with the annotation

$$\lambda_{q' \odot q} = (\lambda_{q' \otimes q}) \upharpoonright_{A \parallel C}$$

We have $(q' \odot q, \lambda_{q' \odot q}) \in \mathbb{T}\text{-Aug}(A \parallel C)$, \mathbb{T} -receptive and \mathbb{T} -courteous.

That $(q' \odot q, \lambda_{q' \odot q})$ has the properties claimed above is easily shown following the inductive definition of $\lambda_{q' \otimes q}$. This is essentially a recast of the proof of lemma 2.3 so we skip it here.

Similarly, extending lemma 4.3, one can follow the proof of proposition 2.5 to inductively prove that:

Lemma 4.18. *Let $q_1 \preceq q'_1 \in \text{Aug}(A^\perp \parallel B)$ and $q_2 \preceq q'_2 \in \mathbb{T}\text{-Aug}(B^\perp \parallel C)$ such that $q_1 = q'_1$, $q_2 = q'_2$ and q_1, q_2 are causally compatible, then q'_1, q'_2 are causally compatible and $q_2 \odot q_1 \preceq q'_2 \odot q'_1$.*

Global renaming The next step is to extend global (and local) renaming that are used in both Strat and Det, in particular to define the tensor product of strategies.

Definition 4.25. Let $f : A \cong A'$ be an isomorphism of event structures and let $q \in \mathbb{T}\text{-Aug}(A)$, the *global renaming of q by f* has underlying augmentation $f * q$, and labeling:

$$\lambda_{f * q} = f.\lambda_q$$

where $f : A \cong A'$ is regarded as a substitution.

It is clear that $(f * q, \lambda_{f * q}) \in \mathbb{T}\text{-Aug}(A')$. Likewise,

Lemma 4.19. *The global renamings on \mathbb{T} -augmentations preserves \mathbb{T} -receptivity, \mathbb{T} -courtesy and \preceq .*

Given two \mathbb{T} -augmentations $\mathfrak{q} \in \mathbb{T}\text{-Aug}(A^\perp \parallel C)$, $\mathfrak{q}' \in \mathbb{T}\text{-Aug}(B^\perp \parallel D)$, their *tensor product* is defined as previously by

$$\mathfrak{q} \otimes \mathfrak{q}' = \gamma * (\mathfrak{q} \parallel \mathfrak{q}')$$

Similarly if $\mathfrak{q} \in \mathbb{T}\text{-Aug}(A^\perp \parallel B)$ and $f : A \cong A'$ is an isomorphism then its *left renaming* is still defined by

$$f \cdot \mathfrak{q} = (f \parallel B) * \mathfrak{q} \in \mathbb{T}\text{-Aug}(A'^\perp \parallel B)$$

Prefixes We conclude this section by extending the notion of prefixes to \mathbb{T} -augmentations, this will be needed in the coming sections to define the rigid \mathbb{T} -strategies and the composition of elementary \mathbb{T} -strategies.

Definition 4.26. Let $\mathfrak{q}, \mathfrak{q}'$ be two \mathbb{T} -augmentations, we say that \mathfrak{q} is a *prefix* of \mathfrak{q}' , written $\mathfrak{q} \hookrightarrow \mathfrak{q}'$, if it holds for the underlying augmentations, and that for every $a \in |\mathfrak{q}|$

$$\lambda_{\mathfrak{q}}(a) = \lambda_{\mathfrak{q}'}(a)$$

Recall that the equality above is the equality on equivalence classes of $\cong_{\mathbb{E}}$ in $\text{Tm}_\Sigma([a])$.

4.3.2 \mathbb{T} -Rig

Based on the previous section, there is not much left in order to define a category of games and rigid \mathbb{T} -strategies, $\mathbb{T}\text{-Rig}$. We set:

Definition 4.27. A *rigid \mathbb{T} -strategy* on a game A is a non-empty and prefix-closed set of \mathbb{T} -courteous and \mathbb{T} -receptive augmentations over A which additionally satisfies receptivity, courtesy from definition 4.5. We write $\sigma : A \xrightarrow{\mathbb{T}\text{-Strat}} B$ for a rigid \mathbb{T} -strategy $\sigma : A^\perp \parallel B$.

The partial order \preceq on \mathbb{T} -augmentations extends to \mathbb{T} -strategies as in definition 4.10.

Example 4.6. The rigid images of concurrent \mathbb{T} -strategies are examples of rigid \mathbb{T} -strategies: for $\sigma : S \rightarrow A$ a concurrent \mathbb{T} -strategy $\text{Rig}(\sigma)$ is again defined as

$$\text{Rig}(\sigma) = \{\mathfrak{q}_x \mid x \in \mathcal{C}(S)\} \quad : \quad A$$

Note that with rigid \mathbb{T} -strategies, two events that map to the same move, share the same causal history but have distinct annotations are distinguished in the model. We say that a concurrent \mathbb{T} -strategy $\sigma : S \rightarrow A$ is *non-deterministically idempotent* if there is $s \neq s'$ in S such that $\sigma(s) = \sigma(s')$, $[s] = [s']$ and $\lambda_S(s) = \lambda_S(s')$.

Compact structure Composition and tensor product of rigid \mathbb{T} -strategies follow component-wise from their definitions on \mathbb{T} -augmentations:

$$\begin{aligned}\tau \odot \sigma &= \{\mathfrak{q}' \odot \mathfrak{q} \mid \mathfrak{q}' \in \tau \ \& \ \mathfrak{q} \in \sigma \text{ causally compatible}\} \\ \sigma \otimes \tau &= \{\mathfrak{q}_\sigma \otimes \mathfrak{q}_\tau \mid \mathfrak{q}_\sigma \in \sigma, \mathfrak{q}_\tau \in \tau\}\end{aligned}$$

By lemma 4.18 and lemma 4.19, using the same reasoning as in the plain case, these operations preserve \preceq on rigid \mathbb{T} -strategies. Similarly, copycat strategies and other structural morphisms are defined as the \mathbb{T} -rigid images of the corresponding concurrent \mathbb{T} -strategies. In the next paragraphs we check that the development in subsection 4.1.2 is still valid so we can state:

Theorem 4.3. *Games and rigid \mathbb{T} -strategies, together with \odot , $(1, \otimes)$ and $(_)^\perp$ form a compact closed category $\mathbb{T}\text{-Strat}$ that is order-enriched over \preceq .*

Primes To prove Theorem 4.3 we extend the proof of subsection 4.1.2 to annotated strategies. We want to show:

$$\begin{array}{ccc} & \text{Rig} & \\ & \curvearrowright & \\ \mathbb{T}\text{-CG} & & \mathbb{T}\text{-Strat} \\ & \curvearrowleft & \\ & \text{top} & \end{array}$$

with $\text{Rig} \circ \text{top} = \text{id}$ and $\text{top} \circ \text{Rig}$ acts as the identity (up to isomorphism) on non-deterministically idempotent concurrent \mathbb{T} -strategies, and the rest of the argument for structure inheritance will be unchanged.

We first precise how the map $\text{top} : \text{Strat} \rightarrow \text{CG}$ extends to \mathbb{T} -strategies

$$\begin{aligned}\text{top} : \mathbb{T}\text{-Strat} &\rightarrow \mathbb{T}\text{-CG} \\ \sigma &\mapsto (\text{Pr}(\sigma), \lambda_{\text{Pr}(\sigma)})\end{aligned}$$

For the causal structure of $\text{Pr}(\sigma)$, definition 4.11 is unchanged. However one must keep in mind that now a prime augmentation \mathfrak{q}_a does not only carry the causal history of its top element a *but also* its annotation function. Similarly, the condition for consistency now also implies that the annotations on the augmentations of a consistent set are all equal (by definition of prefixes). This helps in defining the annotation function of $\text{top}(\sigma)$:

$$\lambda_{\text{top}(\sigma)}(\mathfrak{q}_{a^+}) = \lambda_{\mathfrak{q}_a}(a^+)[\text{Pr}(\mathfrak{q}_a)]$$

where $\text{Pr}(\mathfrak{q}_a) : \mathfrak{q}_a \simeq [\mathfrak{q}_a]_{\text{Pr}(\sigma)}$ is the order-isomorphism defined by $(a' \in \mathfrak{q}_a) \mapsto ([a']_{\mathfrak{q}_a}, \lambda_{\mathfrak{q}_a}(a'))$, that is $(a' \in \mathfrak{q}_a) \mapsto \mathfrak{q}_{a'}$ by definition of prefix. Then, by \mathbb{T} -courtesy of \mathfrak{q}_a , we have that $\lambda_{\text{top}(\sigma)}(\mathfrak{q}_{a^+})$ actually is an element of $\mathbb{T}([\mathfrak{q}_a]_{\text{Pr}(\sigma)}^-)$.

One can thus complete lemma 4.4 and see that:

Lemma 4.20. *For $\sigma : A$ be a rigid \mathbb{T} -strategy and $x \in \mathcal{C}(\text{Pr}(\sigma))$ then*

$$\text{top}(\sigma).(\lambda_{\text{top}(\sigma)})_{\upharpoonright x} = \lambda_{\mathbb{Q}_x}$$

Proof. Let $a \in \mathbb{Q}_x$, then $\text{top}(\sigma)_{\upharpoonright [\mathbb{Q}_a]}$ is the inverse of $\text{Pr}(\mathbb{Q}_a)$ as defined above, so $(\text{top}(\sigma).\lambda_{\text{top}(\sigma)})(a) = \lambda_{\mathbb{Q}_a}(a)$. Furthermore $a \in \mathbb{Q}_x$ implies that $\mathbb{Q}_a \hookrightarrow \mathbb{Q}_x$, so by rigidity $\lambda_{\mathbb{Q}_a}(a) = \lambda_{\mathbb{Q}_x}(a)$, concluding the proof. \square

As a consequence, corollary 4.1, stating that $\text{Rig}(\text{Pr}(\sigma)) = \sigma$, also holds in the annotated case.

Similarly, following lemma 4.5, a simple check on the annotations show that if $\sigma : S \rightarrow A$ is a concurrent \mathbb{T} -strategy that validates non-deterministic idempotence then $\text{top}(\text{Rig}(\sigma)) \cong \sigma$. Indeed, the isomorphism of concurrent strategies:

$$\begin{aligned} \varphi : S &\rightarrow \text{Pr}(\text{Rig}(\sigma)) \\ s &\mapsto \mathbb{Q}_{[s]} = \mathbb{Q}_s \end{aligned}$$

preserves labelling since for $s \in S^+$,

$$\begin{aligned} \lambda_{\text{top}(\text{Rig}(\sigma))}(\mathbb{Q}_s) &= \lambda_{\mathbb{Q}_s}(\sigma(s))[\text{Pr}(\mathbb{Q}_s)] \\ &= \lambda_{\sigma}(s)[\sigma][\text{Pr}(\mathbb{Q}_s)] \\ &= \lambda_{\sigma}(s)[\varphi] \end{aligned}$$

Functoriality of Rig and Rig \circ top That $\text{Rig}(\tau \odot \sigma) = \text{Rig}(\tau) \odot \text{Rig}(\sigma)$ (lemma 4.8) extends to concurrent \mathbb{T} -strategies, is immediate since the check on annotations has already been done in lemma 4.17

Then, for $\text{Rig}(\sigma \otimes \tau) = \text{Rig}(\sigma) \otimes \text{Rig}(\tau)$ (lemma 4.8), one needs to make sure that given $x_S \otimes x_T \in \mathcal{C}(S \parallel T)$, the equality $\mathbb{Q}_{x_S \otimes x_T} = \mathbb{Q}_{x_S} \otimes \mathbb{Q}_{x_T}$ also holds on annotations. This follows from the equalities:

$$\begin{aligned} \lambda_{\mathbb{Q}_{x_S \otimes x_T}} &= (\sigma \otimes \tau).(\lambda_{\sigma \otimes \tau})_{\upharpoonright x_S \otimes x_T} \\ &= (\sigma \otimes \tau).(\lambda_{\sigma} \parallel \lambda_{\tau})_{\upharpoonright x_S \parallel x_T} \\ &= (\gamma \circ (\sigma \parallel \tau)).((\lambda_{\sigma})_{\upharpoonright x_S} \parallel (\lambda_{\tau})_{\upharpoonright x_T}) \\ &= \gamma.(\lambda_{\mathbb{Q}_{x_S}} \parallel \lambda_{\mathbb{Q}_{x_T}}) \end{aligned}$$

Finally, Rig preserves isomorphism on \mathbb{T} -concurrent strategies by definition (hence extending lemma 4.6).

Conversely, for the functoriality of Rig \circ top, the proof for composition in the plain case (lemma 4.10) extends straightforwardly to \mathbb{T} -annotations since the check on annotations has already been done in lemma 4.20.

For the preservation of tensor, extending lemma 4.9 amounts to checking that the isomorphism

$$\begin{aligned} \mathcal{C}(\text{Pr}(\sigma \otimes \tau)) &\cong \mathcal{C}(\text{Pr}(\sigma)) \otimes \mathcal{C}(\text{Pr}(\tau)) \\ \text{Pr}(\mathfrak{q}_\sigma \otimes \mathfrak{q}_\tau) &\mapsto \text{Pr}(\mathfrak{q}_\sigma) \otimes \text{Pr}(\mathfrak{q}_\tau) \end{aligned}$$

preserves annotations. This also follows from lemma 4.20 and checking that $\lambda_{\mathfrak{q}_\sigma \otimes \mathfrak{q}_\tau} = \lambda_{\mathfrak{q}_\tau} \otimes \lambda_{\mathfrak{q}_\sigma}$ which follows exactly the same argument as above.

With this last result, we are done with extending the structure of the proof used in section 4.1.1 to \mathbb{T} -augmentations and can state:

Theorem 4.4. *Games and rigid \mathbb{T} -strategies, together with \odot , $(1, \otimes)$ and $(_)^\perp$ form a compact closed category Strat . It is enriched over \preceq*

4.3.3 \mathbb{T} -Det

As for rigid strategies, elementary strategies extend with \mathbb{T} -annotations in a smooth way.

Definition 4.28. An *elementary \mathbb{T} -strategy* on an elementary game A is a \mathbb{T} -courteous and \mathbb{T} -receptive augmentation $\sigma = (|\sigma|, \leq_\sigma) \in \mathbb{T}\text{-Aug}(A)$ that also verifies receptivity and courtesy from definition 4.14.

We write $\sigma : A \xrightarrow{\mathbb{T}\text{-Det}} B$ for an elementary \mathbb{T} -strategy $\sigma : A^\perp \parallel B$.

According to lemma 4.16, that relates annotations on \mathbb{T} -courteous and \mathbb{T} -receptive augmentations with annotations on \mathbb{T} -strategies, one can still view elementary \mathbb{T} -strategies as concurrent \mathbb{T} -strategies through the identity-on-events map of event structures.

This also implies that lemma 4.12 remains valid: if two elementary \mathbb{T} -strategies are isomorphic in $\mathbb{T}\text{-CG}$ then they are actually equal.

As previously, examples of elementary \mathbb{T} -strategies are the copycat \mathbb{T} -strategies on elementary games or the unit and co-unit strategies η_A, ν_A .

Composition. Recall that the interaction of two elementary strategies $\sigma : A \xrightarrow{\text{Det}} B$ and $\tau : B \xrightarrow{\text{Det}} C$ is given by

$$\tau \circledast \sigma = (\sigma \parallel C) \wedge (A \parallel \tau) \in \text{Aug}(A \parallel B \parallel C)$$

By proposition 4.3 this also corresponds to the maximal interaction $\mathfrak{q}_\tau \circledast \mathfrak{q}_\sigma$ in $\text{Fat}(\sigma) \circledast \text{Fat}(\tau)$. Following definition 4.23 and the fact that, by rigid inclusion, $(\lambda_\sigma)_{\upharpoonright \mathfrak{q}_\sigma} = \lambda_{\mathfrak{q}_\sigma}$ and $(\lambda_\tau)_{\upharpoonright \mathfrak{q}_\tau} = \lambda_{\mathfrak{q}_\tau}$ we simply set

Definition 4.29. Let $\sigma : A \xrightarrow{\mathbb{T}\text{-Det}} B$ and $\tau : B \xrightarrow{\mathbb{T}\text{-Det}} C$ be elementary \mathbb{T} -strategies, their *interaction* is $\tau \circledast \sigma \in \text{Aug}(A \parallel B \parallel C)$ together with annotations

$$\begin{aligned} \lambda_{\tau \circledast \sigma} : \quad & |\tau \circledast \sigma| && \rightarrow & \text{Tm}_\sigma(|\tau \circledast \sigma|) \\ & e \in (A^\perp \parallel B^+) && \mapsto & \lambda_\sigma(e)[\lambda_{\tau \circledast \sigma}] \\ & e' \in (B^- \parallel C) && \mapsto & \lambda_\tau(e')[\lambda_{\tau \circledast \sigma}] \end{aligned}$$

By lemma 4.16, this annotation is in line with the isomorphism of event structures $\tau \circledast \sigma \cong \tau \circledast_{\text{CG}} \sigma$ proved in section 4.2. Hence, keeping $\tau \odot \sigma = (\tau \circledast \sigma)_{\downarrow(A \parallel C)}$ for the composition of elementary \mathbb{T} -strategies, the isomorphism $\tau \odot \sigma \simeq \tau \odot_{\text{CG}} \sigma$ is preserved in $\mathbb{T}\text{-CG}$, and we have

$$\tau \odot \sigma \simeq \tau \odot_{\mathbb{T}\text{-CG}} \sigma$$

for $\odot_{\mathbb{T}\text{-CG}}$ the usual composition in $\mathbb{T}\text{-CG}$.

Following the same reasoning as in section 4.2, $\mathbb{T}\text{-Det}$ thus inherits the categorical structure of $\mathbb{T}\text{-CG}$, hence:

Proposition 4.3. *Elementary games and \mathbb{T} -strategies define a category, written $\mathbb{T}\text{-Det}$.*

Tensor The *tensor product* of two elementary \mathbb{T} -strategies remains

$$\sigma \otimes \tau = \gamma * (\sigma \parallel \tau) : (A \otimes B)^\perp \parallel (C \otimes D)$$

for $\sigma : A \xrightarrow{\mathbb{T}\text{-Det}} C$, $\tau : B \xrightarrow{\mathbb{T}\text{-Det}} D$ and $\gamma : (A \parallel C) \parallel (B^\perp \parallel D) \cong (A^\perp \parallel B^\perp) \parallel (C \parallel D)$. Following lemma 4.19 the construction above is well-defined. Hence one can replay the same trick as in section 4.2 to prove the bifactoriality of \otimes and to define the structural strategies of the corresponding monoidal structure from local renaming of copycats. Indeed, lemma 4.15, stating that the renaming of an elementary strategy via an isomorphism f is isomorphic in CG to that same strategy post-composed by f , extends straightforwardly to elementary \mathbb{T} -strategies.

Order enrichment The partial order \preceq on \mathbb{T} -augmentations from definition 4.21 defines a partial order on elementary \mathbb{T} -strategies. From lemma 4.19, it is direct that \otimes on elementary \mathbb{T} -strategies preserves \preceq . A little more involved, the preservation of \preceq by composition is inherited from the order enrichment of Det (below theorem 4.2) and, by proposition 2.5, following the correspondence between the composition in $\mathbb{T}\text{-Det}$ and the composition in $\mathbb{T}\text{-CG}$.

In the end,

Theorem 4.5. *Elementary games and elementary \mathbb{T} -strategies, together with \odot , $(1, \otimes)$ and $(_)\perp$ form a compact closed category $\mathbb{T}\text{-Det}$ that is enriched over \preceq .*

Note that this theorem is also a direct consequence of the categorical structure of $\mathbb{T}\text{-Strat}$ (theorem 4.3) and of its correspondence with $\mathbb{T}\text{-Det}$, which is a straightforward extension of corollary 4.3 to \mathbb{T} -annotations.

PART II

Term-strategies for Herbrand's theorem

The search for a compositional Herbrand theorem.

In this part we present an extended version of the work carried out in [ACHW18]. This work is about giving a compositional interpretation of the foundational theorem of Herbrand for first order classical logic [Her30, Bus94].

In its simplest form, Herbrand’s theorem reduces the validity of a first-order purely existential formula, $\exists x.\varphi$ with φ quantifier free, to that of a finite disjunction: $\exists x.\varphi$ is valid iff one can find a *finite* collection of terms t_1, \dots, t_n , such that $\bigvee \varphi(t_i)$ is a propositional tautology. These terms are called *Herbrand witnesses* for the formula $\exists x.\varphi$. Although not discovered this way, this simple form of Herbrand theorem is a direct consequence of completeness and compactness.

Herbrand’s result can be extended to general formulas. A common way to do so is by reduction to the purely existential case: a formula is converted to prenex normal form, and universally quantified variables are replaced with new function symbols added to the signature of the theory, in a process called *Herbrandization* [Bus94] (dual to Skolemization).

However, Herbrand witnesses obtained this way are not composable: in general, given witnesses for $\vdash A$ and $\vdash A \implies B$ there is a priori no direct way to deduce witnesses for $\vdash B$ [Koh99]. Understanding how the data of these witnesses can be elaborated to allow such a composition has thus become) has thus become a question of interest in proof theory, in particular as a way to design alternative proof formalism [Hei10, McK13, HW13].

Indirectly, proof mining techniques such as functional interpretations provide a way to extract Herbrand witnesses compositionally [GK05]. Although they interpret cuts in proofs (and are compositional in essence), functional interpretations make no pretence to be faithful to the structure of proofs as encapsulated in classical sequent calculus: they explore in a sequential order terms proposed by a proof as witnesses for existential quantifiers, but this order is certainly not intrinsic to the proof. From a proof-theoretic perspective – and closer to the original spirit of Herbrand’s theorem –, it is thus natural to seek a compositional form of Herbrand’s Theorem faithful to the structure of proofs and to the dependencies between terms.

For cut-free proofs, Miller’s *expansion trees* [Mil87] – a modern view on Herbrand witnesses – capture precisely this “Herbrand content” (the information pertaining to quantifier instantiations), but, as the original Herbrand witnesses, they lack compositionality. Yet, in an other approach to the former question, recent works have sought generalisations of these trees that support cuts. These include Heijltjes’ *proof forests* [Hei10], McKinley’s *Herbrand nets* [McK13], and Hetzl and Weller’s more recent *expansion trees with*

cuts [HW13]. In all three cases, a generalisation of expansion trees allowing explicit cuts is given along with a weakly normalising cut reduction procedure, proved correct via syntactic means.

In spirit, our work is close to this recent line of work but with a major shift of perspective: by making explicit the game-theoretic ideas that underlies the literature on expansion trees, we approach the problem of finding a compositional structure for them semantically rather than syntactically. More precisely, we embed expansion trees in the realm of elementary Σ -strategies, which are *by design* compositional, and compute them by directly interpreting first order classical proofs in this game model.

Beyond the term information, the key ingredient of this model is the causal structure of strategies that allows us to represent transparently the dependencies between quantifiers implicitly carried by sequent proofs. Were we interested only in cut-free sequent calculus our strategies would essentially be Miller’s expansion trees, but enriched with *explicit acyclicity witnesses*. This additional data makes the process of composition easier.

Related work This work fits in the longstanding active topic of the computational content of classical logic, with a wealth of related works.

Roughly speaking, there are two families of approaches on that matter. On the one hand, some (including the functional interpretations mentioned above) extract from proofs a sequential procedure, *e.g.* via translation to sequential calculi or by annotating a proof to sequentialize or determinize its behaviour under cut reduction [Gir91, DJS97]. Other than that cited above, influential developments in this “polarized” approach include work by Berardi [BB94], Coquand [Coq95], Parigot [Par92], Krivine [Kri09], and others. Polarization yields better-behaved dynamics and a non-degenerate equational theory but distorts the intent of the proof by an added unintended sequentiality: two proofs that differ only by the order of introduction of two consecutive existential quantifiers will be distinguished in those models. On the game-theoretic front of this line of work, our model is closely related to Laurent’s model for the first-order $\lambda\mu$ -calculus [Lau10] and also related to Mimram’s categorical construction of a games model for a linear first-order logic without propositional connectives [Mim11].

On the other hand, some works avoid polarization – including, of course, Gentzen’s *Hauptsatz* [Gen35]. This causes issues, notably unrestricted cut reduction yields a degenerate equational theory [Gir91] and enjoys only *weak*, rather than *strong*, normalization [DJS97]. Nevertheless, witness extraction remains possible (though it is non-deterministic). Particularly relevant to

our endeavour is the recent activity around the matter of enriching expansion trees so as to support cuts as mentioned above [Hei10, McK13, HW13].

Outline In chapter 5 we recall first order logic, Herbrand’s theorem and expansion trees. We also introduce the precise game models in which expansion trees can be presented as Σ -strategies – a model in which games have *winning conditions* –, leading to a first reformulation of Herbrand theorem.

The two other chapters describe the interpretation of first order proofs as winning strategies: In chapter 6 we give the interpretation of propositional multiplicative linear logic (MLL) by exhibiting a *-autonomous structure in our model, then deal with linear quantifiers, yielding a lax model of first order MLL. In chapter 7 we add contraction and weakening and complete the interpretation of LK. From this we derive our compositional version of Herbrand’s theorem and discuss some of the computational features of the LK sequent calculus reflected in our model.

From Herbrand proofs to winning strategies

“What more do we know when we have proved a theorem by restricted means than if we merely know it is true?”

Kreisel’s question is the driving force for much modern Proof Theory. It is especially critical for first order classical logic which, contrary to intuitionistic logic is known to be non-constructive. In particular, classical logic fails to have the *witness* property: if $\exists x \varphi$ holds then one might not be able to give a *single* term t such that $\varphi(t)$ holds. Consider the formula

$$\exists x (P(x) \implies P(f(x))) \tag{5.1}$$

it is valid (provided the language has some constant symbol c), but there is no first-order term t such that $P(t) \implies P(f(t))$ holds.

In his thesis however [Her30], Herbrand proved that, although no single closed term can serve as a witness of a formula $\exists x \varphi(x)$ (with φ quantifier-free), there always exists *finitely* many terms t_1, \dots, t_n such that $\varphi(t_1) \vee \dots \vee \varphi(t_n)$ holds. The single witness is replaced with a finite disjunction and the extraction of such t_i s is widely regarded as an early account of the computational content of classical proofs.

Before going any further on this statement, we remind some basics on first order classical logic and set notations. A reader familiar with this topic may move directly to the next two sections in which we further introduce Herbrand’s theorem, giving its general form via expansion trees, and then present how these trees can be viewed as concurrent strategies in the framework of elementary games and Σ -strategies.

5.1 Preliminaries on LK_1

5.1.1 First order classical formulas

First order formulas are syntactic objects defined from: a *first order signature* Σ as defined in section 2.1.1 – for simplicity we will furthermore assume that

it contains at least one constant symbol c –; a countable set of *predicate symbols* (ranging over \mathbf{P}, \mathbf{Q} , etc.) which also come equipped with an arity $n \in \mathbb{N}$; and the usual propositional constant, connectives and first order quantifiers which we shall soon recall.

Letting \mathcal{V} be a set of *variable names*, the *atomic formulas over \mathcal{V}* correspond to the propositional constants \top (true) and \perp (false) together with *literals*:

Definition 5.1. A *literal* with free variables in \mathcal{V} has the form $\mathbf{P}(t_1, \dots, t_n)$ or $\neg\mathbf{P}(t_1, \dots, t_n)$, where \mathbf{P} is a n -ary predicate symbol and $t_i \in \text{Tm}_\Sigma(\mathcal{V})$.

We denote the set of literals with free variables in \mathcal{V} by $\text{Lit}_\Sigma(\mathcal{V})$.

In our model construction, we will use *infinitary quantifier-free formulas*, that are, atomic formulas closed under *countable conjunction* (\wedge) and *disjunction* (\vee). We introduce them together with their finitary version:

Definition 5.2. The *infinitary quantifier-free formulae* with variables in \mathcal{V} are given by the grammar below.

$$\begin{aligned} \varphi, \psi ::= & \mathbf{P}(t_1, \dots, t_n) \mid \neg\mathbf{P}(t_1, \dots, t_n) \quad (\in \text{Lit}_\Sigma(\mathcal{V})) \\ & \perp \mid \top \mid \bigwedge_{i \in I} \varphi_i \mid \bigvee_{i \in I} \psi_i \end{aligned}$$

where I is any at most countable set.

We write $\text{QF}_\Sigma^\infty(\mathcal{V})$ for the set of infinitary quantifier-free formulas on set of variables \mathcal{V} and $\text{QF}_\Sigma(\mathcal{V})$ for the corresponding set of *finite* formulas, where all disjunctions and conjunctions are finite.

Finally, *first order formulas* are quantifier free formulas closed under *existential*, \exists , and *universal*, \forall , quantifiers:

Definition 5.3. The set of *first-order formulas* with variable in \mathcal{V} , written $\text{Form}_\Sigma(\mathcal{V})$ is generated by:

$$\begin{aligned} \varphi, \psi \in \text{Form}_\Sigma(\mathcal{V}) ::= & \square x. \varphi \quad (\square \in \{\forall, \exists\}, \varphi \in \text{Form}_\Sigma(\mathcal{V} \cup \{x\})) \\ & \mid \varphi \quad (\in \text{QF}_\Sigma(\mathcal{V})) \end{aligned}$$

Formulas are considered up to α -conversion and assumed to satisfy Barendregt's convention (*i.e.* all bound variables are chosen to be different from the free variables).

Given a substitution $\gamma : \mathcal{V}_1 \xrightarrow{\Sigma} \mathcal{V}_2$, the substitution defined on terms extends to formulas with capture avoidance implicitly handle by Barendregt's

convention:

$$\begin{aligned}
\perp[\gamma] &= \perp \\
\top[\gamma] &= \top \\
(\varphi \wedge \psi)[\gamma] &= \varphi[\gamma] \wedge \psi[\gamma] \\
(\varphi \vee \psi)[\gamma] &= \varphi[\gamma] \vee \psi[\gamma] \\
(P(t_1, \dots, t_n))[\gamma] &= P(t_1[\gamma], \dots, t_n[\gamma]) \\
(\neg P(t_1, \dots, t_n))[\gamma] &= \neg P(t_1[\gamma], \dots, t_n[\gamma]) \\
(\forall x. \varphi)[\gamma] &= \forall y. \varphi[y/x][\gamma] \\
(\exists x. \varphi)[\gamma] &= \exists y. \varphi[y/x][\gamma]
\end{aligned}$$

Finally, the *logical negation* is not considered a logical connective: the negation φ^\perp of φ is obtained by induction on formulas following the De Morgan rules:

$$\begin{aligned}
\perp^\perp &= \top \\
\top^\perp &= \perp \\
(\varphi \wedge \psi)^\perp &= \varphi^\perp \vee \psi^\perp \\
(\varphi \vee \psi)^\perp &= \varphi^\perp \wedge \psi^\perp \\
(P(t_1, \dots, t_n))^\perp &= \neg P(t_1, \dots, t_n) \\
(\neg P(t_1, \dots, t_n))^\perp &= P(t_1, \dots, t_n) \\
(\forall x. \varphi)^\perp &= \exists x. \varphi^\perp \\
(\exists x. \varphi)^\perp &= \forall x. \varphi^\perp
\end{aligned}$$

As it does not affect the binding structure, it is clear that the negation of a formula is a formula. Similarly, the *logical implication* $\varphi \implies \psi$ used in the introduction is not a proper connective, but rather a shortening for $\varphi^\perp \vee \psi$.

Validity. The above definitions for negation and implication follow from the formal Tarskian semantics on classical formulas, which we briefly recall. First, one needs to choose a *model*, that is, a Σ -algebra $(D, \llbracket - \rrbracket)$ (see definition p 57) together with an interpretation $\llbracket P \rrbracket : D^n \rightarrow \{\perp, \top\}$ for every predicate P of arity n . Then,

Definition 5.4. For $\varphi \in \text{Form}_\Sigma(\mathcal{V})$ and $\rho \in D^\mathcal{V}$ is a \mathcal{V} -valuation, the *evaluation of φ* following the (D, ρ) -Tarskian semantics, written $\llbracket \varphi \rrbracket_D^\rho$, is given by:

$$\begin{aligned}
\llbracket \top \rrbracket_D^\rho &= \top \\
\llbracket \perp \rrbracket_D^\rho &= \perp \\
\llbracket P(t_1, \dots, t_n) \rrbracket_D^\rho &= \llbracket P \rrbracket(\llbracket t_1 \rrbracket \rho, \dots, \llbracket t_n \rrbracket \rho) \\
\llbracket \neg P(t_1, \dots, t_n) \rrbracket_D^\rho &= \neg \llbracket P \rrbracket(\llbracket t_1 \rrbracket \rho, \dots, \llbracket t_n \rrbracket \rho)
\end{aligned}$$

$$\begin{aligned}
\llbracket \varphi \wedge \psi \rrbracket_D^\rho &= \begin{cases} \top & \text{if } \llbracket \varphi \rrbracket_D^\rho = \llbracket \psi \rrbracket_D^\rho = \top \\ \perp & \text{otherwise} \end{cases} \\
\llbracket \varphi \vee \psi \rrbracket_D^\rho &= \begin{cases} \perp & \text{if } \llbracket \varphi \rrbracket_D^\rho = \llbracket \psi \rrbracket_D^\rho = \perp \\ \top & \text{otherwise} \end{cases} \\
\llbracket \forall x. \varphi \rrbracket_D^\rho &= \begin{cases} \top & \text{if for all } d \in D \llbracket \varphi \rrbracket_D^{\rho \uplus x \mapsto d} = \top \\ \perp & \text{otherwise} \end{cases} \\
\llbracket \exists x. \varphi \rrbracket_D^\rho &= \begin{cases} \top & \text{if for some } d \in D, \llbracket \varphi \rrbracket_D^{\rho \uplus x \mapsto d} = \top \\ \perp & \text{otherwise} \end{cases}
\end{aligned}$$

This valuation is also defined on infinitary quantifier-free formulas, keeping the first four lines and setting

$$\begin{aligned}
\llbracket \bigwedge_{i \in I} \varphi_i \rrbracket_D^\rho &= \begin{cases} \top & \text{if for all } i \in I, \llbracket \varphi_i \rrbracket_D^\rho = \top \\ \perp & \text{otherwise} \end{cases} \\
\llbracket \bigvee_{i \in I} \varphi_i \rrbracket_D^\rho &= \begin{cases} \top & \text{if for some } i \in I, \llbracket \varphi_i \rrbracket_D^\rho = \top \\ \perp & \text{otherwise} \end{cases}
\end{aligned}$$

for the valuation of countable conjunction and disjunction. It is worth noticing that, these are very close to the respective universal and existential valuations.

Using the above semantics, we can define *tautologies*:

Definition 5.5. Let $\varphi \in \text{Form}(\mathcal{V})$, we say that φ is a *tautology* iff for all model D and \mathcal{V} -valuation $\rho \in D^\mathcal{V}$, we have $\llbracket \varphi \rrbracket_D^\rho = \top$. And similarly for $\varphi \in \text{QF}_\Sigma^\infty(\mathcal{V})$ a (potentially) infinite quantifier-free formula.

We write $\models \varphi$ to indicate that φ is a tautology.

One foundational theorem in proof theory is Gödel completeness theorem that establishes a correspondence between finite classical tautologies and *provable formulas* in the *classical sequent calculus* LK_1 which we now describe.

5.1.2 Classical sequent calculus

The sequent calculus has been introduced by Gentzen [Gen35] as a formal way to describe mathematical proofs.

A *sequent* is a judgment of the form

$$\vdash^\mathcal{V} \varphi_1, \dots, \varphi_n$$

with, for all $1 \leq i \leq n$, $\varphi_i \in \text{Form}_\Sigma(\mathcal{V})$. Note that unlike what is commonly done in proof theory, we keep track explicitly of the available free variables in the sequent. In particular, we will be unable to instantiate existential

\mathcal{V} -MLL	
$\text{Ax} \frac{}{\vdash^{\mathcal{V}} \varphi^\perp, \varphi} \text{fv}(\varphi) \subseteq \mathcal{V}$	$\text{CutT} \frac{\vdash^{\mathcal{V}} \Gamma, \varphi \quad \vdash^{\mathcal{V}} \varphi^\perp, \Delta}{\vdash^{\mathcal{V}} \Gamma, \Delta}$
$\text{Ex} \frac{\vdash^{\mathcal{V}} \Gamma, \varphi, \psi, \Delta}{\vdash^{\mathcal{V}} \Gamma, \psi, \varphi, \Delta}$	$\top\text{I} \frac{}{\vdash^{\mathcal{V}} \top} \quad \perp\text{I} \frac{\vdash^{\mathcal{V}} \Gamma}{\vdash^{\mathcal{V}} \Gamma, \perp}$
$\wedge\text{I} \frac{\vdash^{\mathcal{V}} \Gamma, \varphi \quad \vdash^{\mathcal{V}} \psi, \Delta}{\vdash^{\mathcal{V}} \Gamma, \varphi \wedge \psi, \Delta}$	$\vee\text{I} \frac{\vdash^{\mathcal{V}} \Gamma, \varphi, \psi, \Delta}{\vdash^{\mathcal{V}} \Gamma, \varphi \vee \psi, \Delta}$

First-order MLL (MLL_1)	LK_1
$\forall\text{I} \frac{\vdash^{\mathcal{V} \cup \{x\}} \Gamma, \varphi}{\vdash^{\mathcal{V}} \Gamma, \forall x. \varphi} x \notin \text{fv}(\Gamma)$	$\text{C} \frac{\vdash^{\mathcal{V}} \Gamma, \varphi, \varphi}{\vdash^{\mathcal{V}} \Gamma, \varphi}$
$\exists\text{I} \frac{\vdash^{\mathcal{V}} \Gamma, \varphi[t/x]}{\vdash^{\mathcal{V}} \Gamma, \exists x. \varphi} t \in \text{Tm}_\Sigma(\mathcal{V})$	$\text{W} \frac{\vdash^{\mathcal{V}} \Gamma}{\vdash^{\mathcal{V}} \Gamma, \varphi}$

Figure 5.1: Rules for the sequent calculus LK_1

quantifiers using witnesses with free variables not in \mathcal{V} . This restriction will not impact the power of the general calculus but will provide useful information when seeking a denotation of proofs in our games model.

We will simply write $\vdash \Gamma$ for $\vdash^\emptyset \Gamma$.

Proofs. The *proofs* are inductively defined using the derivation rules listed on figure 5.1. Apart from the explicit set \mathcal{V} of free variables, these define a rather standard one-sided sequent calculus with rules presented in the multiplicative style.

Note that instead of the usual presentation of rules into the four Identity (Ax, CutT), Structural (C, W, Ex), Propositional ($\top\text{I}$, $\wedge\text{I}$, $\vee\text{I}$), and Quantifier ($\forall\text{I}$, $\exists\text{I}$) groups, we chose to organise rules according to the various proof systems they define: the whole system corresponds to LK_1 ; removing the *contraction*, C, and *weakening*, W, yields MLL_1 , the proof system for *first order multiplicative linear logic* (although the present \wedge and \vee are then usually denoted \otimes and \wp); finally, removing the two quantifier introduction rules, $\forall\text{I}$ and $\exists\text{I}$, leads to \mathcal{V} -MLL, the multiplicative linear logic on $\text{QF}_\Sigma(\mathcal{V})$.

$$\begin{array}{c}
\frac{}{\vdash^{\{y\}} P(y), \neg P(y)} \text{Ax} \\
\frac{\vdash^{\{y\}} \neg P(s), P(y), \neg P(y), (\forall z. P(z))}{\vdash^{\{y\}} \neg P(s), P(y), \neg P(y) \vee (\forall z. P(z))} \text{W} \\
\frac{\vdash^{\{y\}} \neg P(s), P(y), \neg P(y) \vee (\forall z. P(z))}{\vdash^{\{y\}} \neg P(s), P(y), \exists x. \neg P(x) \vee (\forall y. P(y))} \text{VI} \\
\frac{\vdash^{\{y\}} \neg P(s), P(y), \exists x. \neg P(x) \vee (\forall y. P(y))}{\vdash \neg P(s), \forall y. P(y), \exists x. \neg P(x) \vee (\forall y. P(y))} \exists_I, x := y \\
\frac{\vdash \neg P(s), \forall y. P(y), \exists x. \neg P(x) \vee (\forall y. P(y))}{\vdash \neg P(s) \vee (\forall y. P(y)), \exists x. \neg P(x) \vee (\forall y. P(y))} \forall I \\
\frac{\vdash \neg P(s) \vee (\forall y. P(y)), \exists x. \neg P(x) \vee (\forall y. P(y))}{\vdash \exists x. \neg P(x) \vee (\forall y. P(y)), \exists x. \neg P(x) \vee (\forall y. P(y))} \exists_I, x := c \\
\frac{\vdash \exists x. \neg P(x) \vee (\forall y. P(y)), \exists x. \neg P(x) \vee (\forall y. P(y))}{\vdash \exists x. \neg P(x) \vee (\forall y. P(y))} \text{C}
\end{array}$$

Figure 5.2: A classical proof for DF

We do not expand further on these systems for now but they will provide us with an incremental way of interpreting LK_1 in our games model.

Figure 5.2 presents a proof derivation for a typical classical first order formula named the *drinker's formula* or the *drinker's paradox*

$$\exists x. P(x) \implies (\forall y. P(y))$$

so called as its logical meaning can be illustrated as follows: in a non empty room, we can always find someone (x) such that if that person is drinking ($P(x)$) then everybody else is drinking as well ($\forall y. P(y)$). Note that when representing proofs, we will in general keep implicit the uses of the exchange rule (EX). However, it is important in the interpretation to keep in mind that contexts really are lists of formulas rather than sets or multisets, and that formally the exchange rule has to be explicitly used.

Similarly, we will implicitly make use of the following admissible rule on free variables when writing down proof trees:

$$\text{W-VAR} \frac{\vdash^{\mathcal{V}} \Gamma}{\vdash^{\mathcal{V} \uplus \{x\}} \Gamma}$$

Yet, in the interpretation, this weakening rule will be explicitly shown.

We say that a sequent $\vdash^{\mathcal{V}} \Gamma$ is *provable* if one can find a proof derivation that ends on $\vdash^{\mathcal{V}} \Gamma$. Similarly, we say that a formula $\varphi \in \text{Form}(\mathcal{V})$ is *provable* if $\vdash^{\mathcal{V}} \varphi$ is provable. As mentioned earlier Gödel completeness theorem states:

Theorem 5.1. *For any closed formula φ , $\models \varphi$ iff φ it is provable in LK_1 .*

$$\begin{array}{c}
\text{CUT} \frac{\frac{\pi_1}{\frac{\forall I}{\frac{\vdash^{\mathcal{V}\Psi\{x\}} \Gamma, \varphi} \vdash^{\mathcal{V}} \Gamma, \forall x. \varphi}} \quad \frac{\pi_2}{\frac{\exists I}{\frac{\vdash^{\mathcal{V}} \varphi^\perp[t/x], \Delta} \vdash^{\mathcal{V}} \exists x. \varphi^\perp, \Delta}}}{\vdash^{\mathcal{V}} \Gamma, \Delta} \rightsquigarrow_{\forall/\exists} \text{CUT} \frac{\frac{\pi_1[t/x]}{\frac{\vdash^{\mathcal{V}} \Gamma, \varphi[t/x]} \vdash^{\mathcal{V}} \Gamma, \Delta} \quad \frac{\pi_2}{\frac{\vdash^{\mathcal{V}} \varphi^\perp[t/x], \Delta} \vdash^{\mathcal{V}} \varphi^\perp[t/x], \Delta}}{\vdash^{\mathcal{V}} \Gamma, \Delta} \\
\\
\text{CUT} \frac{\frac{\pi_1}{\frac{\vdash^{\mathcal{V}} \Gamma, \psi} \vdash^{\mathcal{V}} \Gamma, \psi} \quad \frac{\pi_2}{\frac{\forall I}{\frac{\frac{\vdash^{\mathcal{V}\Psi\{x\}} \psi^\perp, \Delta, \varphi} \vdash^{\mathcal{V}} \psi^\perp, \Delta, \forall x. \varphi} \vdash^{\mathcal{V}} \Gamma, \Delta, \forall x. \varphi}}}{\vdash^{\mathcal{V}} \Gamma, \Delta, \forall x. \varphi} \rightsquigarrow_{\text{CUT}/\forall} \text{CUT} \frac{\frac{\pi_1}{\frac{\vdash^{\mathcal{V}\Psi\{x\}} \Gamma, \psi} \vdash^{\mathcal{V}\Psi\{x\}} \Gamma, \psi} \quad \frac{\pi_2}{\frac{\vdash^{\mathcal{V}\Psi\{x\}} \psi^\perp, \Delta, \varphi} \vdash^{\mathcal{V}\Psi\{x\}} \psi^\perp, \Delta, \varphi}}{\frac{\forall I}{\frac{\vdash^{\mathcal{V}\Psi\{x\}} \Gamma, \Delta, \varphi} \vdash^{\mathcal{V}} \Gamma, \Delta, \forall x. \varphi}} \\
\\
\text{CUT} \frac{\frac{\pi_1}{\frac{\vdash^{\mathcal{V}} \Gamma, \psi} \vdash^{\mathcal{V}} \Gamma, \psi} \quad \frac{\pi_2}{\frac{\exists I}{\frac{\frac{\vdash^{\mathcal{V}} \psi^\perp, \Delta, \varphi[t/x]} \vdash^{\mathcal{V}} \psi^\perp, \Delta, \exists x. \varphi} \vdash^{\mathcal{V}} \Gamma, \Delta, \exists x. \varphi}}}{\vdash^{\mathcal{V}} \Gamma, \Delta, \exists x. \varphi} \rightsquigarrow_{\text{CUT}/\exists} \text{CUT} \frac{\frac{\pi_1}{\frac{\vdash^{\mathcal{V}} \Gamma, \psi} \vdash^{\mathcal{V}} \Gamma, \psi} \quad \frac{\pi_2}{\frac{\vdash^{\mathcal{V}} \psi^\perp, \Delta, \varphi[t/x]} \vdash^{\mathcal{V}} \psi^\perp, \Delta, \varphi[t/x]}}{\frac{\exists I}{\frac{\vdash^{\mathcal{V}} \Gamma, \Delta, \varphi[t/x]} \vdash^{\mathcal{V}} \Gamma, \Delta, \exists x. \varphi}}
\end{array}$$

Figure 5.3: Additional cut elimination rules for MLL_1

Cut reduction In LK_1 as in $MLL(1)$, the CUT rule is admissible, meaning that the set of provable formulas in the systems with and without CUT are the same (the set of proofs for these formulas of course differ). In other words the above theorem can be rephrased as follows

Theorem 5.2. *For any closed formula φ , $\vdash \varphi$ iff φ it has a cut-free proof in LK .*

Usually, such a theorem is proved by inductively eliminating cuts from a given proof, following a *cut reduction* procedure. The following theorem, on cut reduction for MLL , is folklore.

Theorem 5.3. *There is a set of reduction rules on MLL sequent proofs, written $\rightsquigarrow_{\text{MLL}}$, such that for any proof π of a sequent $\vdash \Gamma$, there is a cut-free π' of Γ such that $\pi \rightsquigarrow_{\text{MLL}}^* \pi'$.*

The reduction $\rightsquigarrow_{\text{MLL}}$ comprises *logical* reductions, reducing a cut on a formula φ/φ^\perp , between two proofs starting with the introduction rule for the main connective of φ/φ^\perp ; and *structural* reductions, consisting in commutations between rules so as to reach the logical steps. We do not detail these reductions further as we will not explicitly study them in the sequel. Instead, we depict on figure 5.3 the additional reduction rules for MLL_1 : a new *logical* reduction (\forall/\exists), and two *structural* reductions for the propagation of cuts past introduction rules for \forall and \exists .

Note that the first rule of figure 5.3 requires the introduction of *substitution* on proofs: given a proof π for $\vdash^{\mathcal{V}_2} \Gamma$ and a substitution $\gamma : \mathcal{V}_1 \rightarrow \mathcal{V}_2$, we set $\pi[\gamma]$ to be a proof of $\vdash^{\mathcal{V}_1} \Gamma[\gamma]$ by applying γ on every formulas and terms in π , propagating it by structural induction. A degenerate case of proof substitution is the substitution of a proof π for $\vdash^{\mathcal{V}} \Gamma$ by the *weakening* $w_{\mathcal{V},x} : \mathcal{V} \uplus \{x\} \rightarrow \mathcal{V}$, leading to $\pi_1[w_{\mathcal{V},x}]$, a proof for $\vdash^{\mathcal{V} \uplus x} \Gamma$. As this leaves the formulas and terms unchanged we leave it implicit in the reduction rules – it is used for instance implicitly in the commutation CUT/ \forall .

Writing $\pi \rightsquigarrow_{\text{MLL}_1} \pi'$ for the reduction obtained with the rules from figure 5.3 together with $\rightsquigarrow_{\text{MLL}}$, the previous theorem extends to:

Theorem 5.4 ([Gir87]). *Let π be any MLL_1 proof of $\vdash^{\mathcal{V}} \Gamma$. Then, there is a cut-free proof π' of $\vdash^{\mathcal{V}} \Gamma$ s.t. $\pi \rightsquigarrow_{\text{MLL}_1}^* \pi'$.*

The classical sequent calculus LK_1 also admits a set of cut reduction rules. These are the one of $\rightsquigarrow_{\text{MLL}_1}$ together with the two “logical” reductions (W, C) and the two structural reductions (W/CUT, C/CUT) depicted on figure 5.4. Writing $\pi \rightsquigarrow_{\text{LK}_1} \pi'$ for the reduction obtained with these rules we again have:

Theorem 5.5. *Let π be any LK_1 proof of $\vdash^{\mathcal{V}} \Gamma$. Then, there is a cut-free proof π' of $\vdash^{\mathcal{V}} \Gamma$ s.t. $\pi \rightsquigarrow_{\text{LK}_1}^* \pi'$.*

Through the *Curry-Howard correspondence*, proofs reductions can be viewed as computations leading to *normal forms*, i.e. cut-free proofs. The cornerstone of this correspondence is to consider cuts as function applications in a typed programming language.

For $\text{MLL}(1)$, the reduction rules define a confluent and strongly normalising rewriting system, making it suitable to work with as a calculus. However, this is *not* the case of $\rightsquigarrow_{\text{LK}_1}$. A typical example of non-confluence is the so called Lafont critical pair:

$$\frac{\frac{\frac{\pi_1}{\vdash \Gamma}}{\vdash \Gamma, A} \text{W} \quad \frac{\frac{\pi_2}{\vdash \Gamma}}{\vdash A^\perp, \Gamma} \text{W}}{\frac{\vdash \Gamma, \Gamma}{\Gamma} \text{C}} \text{CUT}$$

$$\begin{array}{c}
\frac{}{\vdash D(f(c)), \neg D(f(c))} \text{Ax} \\
\frac{}{\vdash \neg D(c), D(f(c)), \neg D(f(c)), D(f(f(c)))} \text{W,W} \\
\frac{}{\vdash \neg D(c) \vee D(f(c)), \neg D(f(c)) \vee D(f(f(c)))} \vee I, \vee I \\
\frac{}{\vdash \neg D(c) \vee D(f(c)), \exists x \neg D(x) \vee D(f(x))} \exists I, x := f(c) \\
\frac{}{\vdash \exists x \neg D(x) \vee D(f(x)), \exists x \neg D(x) \vee D(f(x))} \exists I, x := c \\
\frac{}{\vdash \exists x \neg D(x) \vee D(f(x))} \text{C}
\end{array}$$

Figure 5.5: Classical proof for a purely existential formula

which reduces to both of the following proofs (for \sim some structural equivalence that we will not precise here):

$$\frac{\frac{\pi_1}{\vdash \Gamma}}{\vdash \Gamma, \Gamma} \text{WK} \sim \frac{\pi_1}{\vdash \Gamma} \qquad \frac{\frac{\pi_2}{\vdash \Gamma}}{\vdash \Gamma, \Gamma} \text{WK} \sim \frac{\pi_2}{\vdash \Gamma}$$

hence equalizing any two proofs of the same formulas. The equational theory obtained by transitive and reflexive closure of $\rightsquigarrow_{\text{LK}_1}$ thus collapses to a Boolean algebra, not a very expressive semantics for whoever wants to do programming!

5.2 Herbrand's theorem

As mentioned in the introduction, Herbrand's theorem in its simpler formulation relates the validity of a purely existential formula, to the existence of a finite set of witnesses – that are closed terms over the signature – such that the finite disjunction obtained by instantiating the formula with this collection of terms is a tautology.

Figure 5.5 depicts a proof of the formula from example 5.1. Collecting all the terms used in the proof one may extract the following valid disjunction

$$(P(c) \implies P(f(c))) \vee (P(f(c)) \implies P(f(f(c))))$$

where the existential quantifier has been instantiated with two witnesses c and $f(c)$. We call the disjunction above a Herbrand disjunction and c and $f(c)$ its witnesses.

It is quite clear that this extraction procedure can be performed on any cut free proof of a valid \exists -formula, and, conversely, that every such disjunction for an \exists -formula can be transformed into a cut free proof. This is the simple version of Herbrand's theorem:

Theorem 5.6. *Consider a formula of the form $\psi = \exists x_1 \dots \exists x_n \varphi(x_1, \dots, x_n)$ where φ is quantifier free. Then, $\models \psi$ iff there are closed terms $(t_{i,j})_{1 \leq i \leq p, 1 \leq j \leq n}$ such that*

$$\models \bigvee_{i=1}^p \varphi(t_{i,1}, \dots, t_{i,n})$$

5.2.1 Herbrand proofs

It is possible to give a generalised version of the theorem above that fits for any kind of formulas. A common way to do so is by reduction to the purely existential case: a formula φ is converted to prenex normal form and universally quantified variables are replaced with new function symbols added to Σ , in a process called *Herbrandization* (dual to Skolemization). For instance, the *drinker's formula*:

$$\exists x \forall y \neg P(x) \vee P(y) \quad (DF)$$

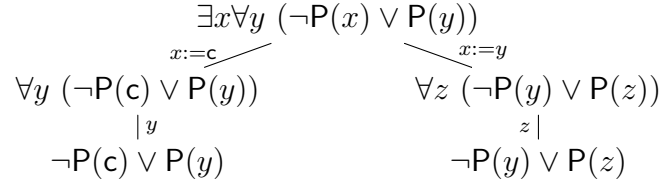
yields by Herbrandization the formula ψ of Example 5.1.

An alternative way – to avoid distortion of the formulas by Herbrandization – is to use more complex structures than just mere sets of terms to extract witnesses. This is necessary *e.g.* to handle the fact that the existential witnesses in the proofs of general first order formulas may depend on universally quantified variables in those formulas. As an example, one can look at the shape of witnesses in the proof of the *DF* formula depicted in figure 5.2.

Generalising disjunctions to a richer structure suited for any first order formulas is the way chosen by Herbrand in his original work. Roughly speaking, he introduced a notion of *Herbrand proof* able to draw a formal correspondence between valid formulas and the propositional tautologies that appear in their cut-free proofs. This lead to:

Theorem 5.7. *For any closed formula φ , $\models \varphi$ iff φ has an Herbrand proof.*

We do not detail this original formalism further here, but rather refer the interested reader to [Bus94] for a complete overview. In the next section however we introduce *expansion trees*, a modern presentation of Herbrand proofs proposed by Miller in [Mil87]. Expansion trees have the advantage of

Figure 5.6: An expansion tree for DF

being more concise and geometric than the original Herbrand proofs, and of having a game semantical flavour. Still, they enjoy the same property:

Theorem 5.8 ([Mil87]). *For any closed formula φ , $\models \varphi$ iff φ has an expansion tree.*

5.2.2 Expansion trees.

As expressed in theorem 5.8, expansion trees are tree structures that witness the validity of classical first order formulas in a way that is close to the structure of the assignment of first-order terms to existential quantifiers, and the causal dependency between quantifiers in (cut-free) proofs of the classical sequent calculus. In this section we do not give a formal definition of expansion trees, instead we introduce them through a game metaphor reminiscent of Coquand's backtracking games for classical arithmetic [Coq95]. This will motivate our own game interpretation of first order classical proofs.

Tree structure On figure 5.6, we depict an *expansion tree* for DF . This tree is rooted in DF and may be read from top to bottom, and from left to right as a game between two players, \exists loïse and \forall bélar, that debate the validity of DF . The rules of the game are as follows: on a formula $\forall x\varphi$, it is \forall bélar's turn to play, he must provide a fresh variable x before the game keeps going on φ ; on $\exists x\varphi$, it is \exists loïse's turn to play, she must provide a *term* t that possibly contains variables previously introduced by \forall bélar, once done, the game keeps going on $\varphi[t/x]$.

On figure 5.6, \exists loïse opens the game and plays c . Then, \forall bélar introduces y , and we reach a position from which neither \exists loïse nor \forall bélar can play anymore – this is a leaf of the tree, a quantifier free formulas.

\exists loïse, though, has a special power: at any time she can *backtrack* to a previous existential position and provide a new witness for it – still having access to the variables played by \forall bélar in between. On figure 5.6, \exists loïse uses this power to backtrack after \forall bélar's move (we jump to the right branch)

and plays y . At last \forall bélaré introduces z and this is a *win* for \exists loïse: the leaves of the tree define a propositional tautology

$$(\neg P(c) \vee P(y)) \vee (\neg P(y) \vee P(z))$$

Validity criterion In the above example, \exists loïse wins: the disjunction of the leaves is a tautology. More generally, every expansion tree defines a quantifier free formula¹ obtained by collecting every of the subformulas explored during the game. This is called the *deep* formula of the expansion tree (as opposed to the formula being proved, referred to as the *shallow* formula in the definition of expansion trees) and is used to referee the game/check that the tree is correct: an expansion tree is valid/ a win for \exists loïse if its deep formula is a *propositional tautology*.

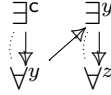
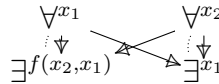
Acyclicity criterion The game metaphor however has limits: it suggests a sequential ordering between branches which expansion trees do not have in reality; they only carry an implicit partial ordering, corresponding to the transitive closure of the structure of the tree together with the variable dependencies of the term annotations. Yet, this ordering is crucial to ensure correctness of the trees. Certainly the tree below should not be valid as the formula at its root is invalid.

$$\begin{array}{c}
 \exists x_1 \forall y_1 P(x_1, y_1) \vee \exists x_2 \forall y_2 \neg P(y_2, x_2) \\
 \swarrow \qquad \searrow \\
 \begin{array}{c}
 \exists x_1 \forall y_1 P(x_1, y_1) \\
 x_1 := y_2 \mid \\
 \forall y_1 P(y_2, y_1) \\
 y_1 \mid \\
 P(y_2, y_1)
 \end{array}
 \qquad
 \begin{array}{c}
 \exists x_2 \forall y_2 \neg P(y_2, x_2) \\
 \mid x_2 := y_1 \\
 \forall y_2 \neg P(y_2, y_1) \\
 \mid y_2 \\
 \neg P(y_2, y_1)
 \end{array}
 \end{array}$$

And indeed, the full definition of expansion trees involves a correctness criterion that forbids this: the causal relation on nodes resulting from the tree structure and its labelling must be *acyclic*. This acyclicity entails the existence of a sequentialization, but committing to one is an arbitrary choice of the above metaphor that is not forced by the proof.

As we will see next, in our representations of proofs (called *winning* (Σ -)strategies), the causal relation on nodes will be made explicit as a partial order. For instance, we display in figure 5.7 the winning Σ -strategy matching,

¹ In comparison with the original definition [Mil87], here expansion trees are considered in their expanded version, meaning that every leaf is a quantifier free formula. This is closer to *e.g.* Hetzl and Weller's *expansion trees with cuts* [HW13].

Figure 5.7: A winning Σ -strategy for DF Figure 5.8: A partially ordered winning Σ -strategy

in our framework, the expansion tree for DF . Another winning Σ -strategy, displayed in figure 5.8, illustrates that this order is not always naturally sequential.

This explicit partial order will in fact lead to a change of perspective on expansion trees: rather than derived afterwards, this order will be considered primitive, and only later decorated with term annotations. Our strategies will thus be more informative than expansion trees and make the acyclicity correctness criterion redundant. This extra information will also help in making expansion trees compositional.

5.3 Games for Herbrand's theorem

5.3.1 Expansion Trees as Winning Σ -Strategies

This section presents our formulation of expansion trees as (elementary) Σ -strategies as introduced in section 2.1.2 and 4.3.3. Although our definitions look superficially different from Miller's, the only fundamental difference will be the explicit display of the dependency between quantifiers of parallel branches.

Trees as Σ -strategies In our interpretation, Σ -strategies will only have events either “ \forall events” or “ \exists events”. Other connectives will not be reflected as moves in the strategies. This is a choice in our semantics to emphasize the role of quantifiers and dependencies between them. Still, we will see later that the propositional connectives are reflected in the structure of our games. Moreover, there is no notion of non-determinism in expansion trees, so our model will be restricted to elementary Σ -strategies.

Figure 5.7 shows the representation as an elementary Σ -strategy of the tree depicted in figure 5.6. Note that here, \forall events are annotated with fresh variables different from their own name, as one could have expect from definition 4.22. This is just a choice of presentation for readability but the actual definition is unchanged. Let us comment on the relation between expansion trees and elementary Σ -strategies.

First of all, note that events in Σ -strategies carry terms, the same way as do universal and existential branches in expansion trees: fresh variables on \forall moves, (possibly open) terms on \exists moves. The definition of Σ -strategies however ensures that these terms respect the explicit causal dependency between quantifiers as the only possible free variables are the ones associated with \forall moves in their causal history. In that sense, Σ -strategies are more general than expansion trees: they have an explicit causal order, and this partial order may be more constraining than the one given by the terms. To capture this difference, we will distinguish *minimal* strategies from others:

Definition 5.6. A elementary Σ -strategy $\sigma : A$ is *minimal* if whenever $a_1 \rightarrow_\sigma a_2^+$ such that $a_1 \notin \text{fv}(\lambda_\sigma(a_2))$, then $a_1 \rightarrow_A a_2$ as well.

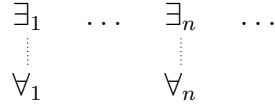
In a minimal Σ -strategy $\sigma : A$, the ordering \leq_σ is actually redundant and can be uniquely recovered from λ_σ and \leq_A .

Formulas as games Σ -strategies will account for first-order *proofs*, and as such, will play on games representing *formulas*. In this chapter we give a first interpretation of formulas as games noted $\llbracket - \rrbracket^\exists$. The existential in superscript emphasizes that in this interpretation, games will be biased towards \exists loïse. Let us make this more precise.

Interpreted formulas will be elementary games: their moves will correspond to the \forall and \exists quantifiers of formulas, partially ordered according to the syntactic structure of these formulas. Every \forall moves will have $-$ polarity (\forall bélar is the Opponent), while \exists moves will have $+$ polarity (\exists loïse is the Player). For this reason, we will often drop the polarity information on \forall and \exists moves.

Similarly to expansion trees where only \exists loïse can replicate her moves (“*backtrack*” – although the terminology is imperfect when strategies are not sequential), games corresponding to formulas in our interpretation will at first be biased towards \exists loïse: each \exists move will exist in as many copies as she might desire, whereas \forall events will not be copied *a priori*.

Figure 5.9 shows the \exists -biased game $\llbracket DF \rrbracket^\exists$ for DF . Although only \exists loïse can replicate her moves, the universal quantifier is also copied as it depends on the existential quantifier. One can check that the strategy depicted on the right of figure 5.7 is a strategy for the game on figure 5.9 – in

Figure 5.9: The elementary game $\llbracket DF \rrbracket^\exists$

fact, in figure 5.7 we also display via dotted lines the immediate dependency of the game.

Note that the receptivity and courtesy conditions on strategies, are consistent with the informal game semantics described in the previous section: receptivity means that \exists loïse cannot refuse to acknowledge a move by \forall bélar, and courtesy that the only new causal constraints that she can enforce with respect to the game is that some existential quantifiers depend on some universal quantifiers.

Winning conditions In order for the model to be able to discriminate valid from invalid strategies, we need to adjoin *winning conditions* to our usual games and define *winning Σ -strategies*. In our interpretation of formulas as games, winning conditions will realise the validity criterion on expansion trees. For that, we decorrelate the syntactic structure of the deep formulas of expansion trees from their instantiation with \exists loïse’s witnesses and \forall bélar’s variables. These “non-instantiated” formulas will be associated with the configurations of the game, while the instantiation will be performed at the level of strategies to check their correctness. We set:

Definition 5.7. A *win-game* is $\mathcal{A} = (A, \mathcal{W}_{\mathcal{A}})$ where A is an elementary game and $\mathcal{W}_{\mathcal{A}}$ defines the *winning conditions* on A :

$$\mathcal{W}_{\mathcal{A}} : (x \in \mathcal{C}^\infty(A)) \rightarrow \mathbf{QF}_\Sigma^\infty(x),$$

for $\mathcal{C}^\infty(A)$ the set of *infinite configurations* of A , that is, the set of (possibly infinite) down-closed subsets $x \subseteq A$ such that for every finite $x' \subseteq x$, $x' \in \text{Con}_A$.

In the above, winning conditions are defined on both finite and infinite configurations as our interpretation of formulas will lead to define infinite games, on which strategies may as well be infinite. We also insist on the fact that winning conditions are *syntactic* objects: there is no valuation attached to them.

A first example of interpretation are the propositional constants and closed literals (*i.e.* $\varphi \in \{\mathbf{P}(t_1, \dots, t_n), \neg \mathbf{P}(t_1, \dots, t_n)\}$ with $t_i \in \text{Tm}_\Sigma(\emptyset)$).

Having no quantifier, they all have base-game \emptyset but differ in their winning condition:

$$\mathcal{W}_{[\perp]^\exists}(\emptyset) = \perp \quad \mathcal{W}_{[\top]^\exists}(\emptyset) = \top \quad \mathcal{W}_{[\varphi]^\exists}(\emptyset) = \varphi$$

We respectively note $\mathbf{1}, \perp, \varphi$ the corresponding wingames. We delay until the next section 5.3.2 the definition of the rest of the interpretation of formulas as (biased) games. However, the idea stays relatively simple: for the game interpreting φ , the winning conditions associate a configuration $x \in \mathcal{C}^\infty(\llbracket \varphi \rrbracket)$ with the deep formula of the expansion tree explored by x in which quantified variables are replaced with the names of the events corresponding to the quantifiers.

Example 5.1. The game *DF* appearing on Figure 5.9 will have winning conditions

$$\begin{aligned} \mathcal{W}_{[DF]^\exists}(\emptyset) &= \perp \\ \mathcal{W}_{[DF]^\exists}(\{\exists_3\}) &= \top \\ \mathcal{W}_{[DF]^\exists}(\{\exists_3, \forall_3\}) &= (\neg P(\exists_3) \vee P(\forall_3)) \\ \mathcal{W}_{[DF]^\exists}(\{\exists_3, \forall_3, \exists_6\}) &= (\neg P(\exists_3) \vee P(\forall_3)) \vee \top \\ \mathcal{W}_{[DF]^\exists}(\{\exists_3, \forall_3, \exists_6, \forall_6\}) &= (\neg P(\exists_3) \vee P(\forall_3)) \vee (\neg P(\exists_6) \vee P(\forall_6)) \\ &\dots \end{aligned}$$

One can see that for interpreted formulas, the winning condition of a configuration x is “syntactically included” in the winning condition of a configuration $x' \supseteq x$. In the above example, the false formula on the first line is due to \exists loïse lost if she does not open the game. Similarly, the true formula on the second and fourth lines are due to \forall bélarde not having played \forall_i yet, yielding victory to \exists loïse on these configurations. Other components correspond to the deep formulas of the corresponding expansion trees.

Winning strategies Winning conditions are syntactic, uninterpreted formulas. In the example above this is illustrated by the fact that we keep the fourth formula as-is although it is equivalent to \top . As mentioned earlier, the evaluation of winning conditions is performed at the level of strategies:

Definition 5.8. Let \mathcal{A} be a win-game and $\sigma : A$ be an elementary Σ -strategy, a configuration $x \in \mathcal{C}^\infty(\sigma)$ is *tautological* in σ if the formula

$$\mathcal{W}_{\mathcal{A}}(x)[\lambda_\sigma]$$

corresponding to the substitution of $\mathcal{W}_{\mathcal{A}}(x) \in \mathbf{QF}_\Sigma^\infty(x)$ by $\lambda_\sigma : x \rightarrow \mathbf{Tm}_\Sigma(x)$, is a (possibly infinite) tautology.

From there we can define winning strategies.

Definition 5.9. Let \mathcal{A} be a win-game and $\sigma : A$ be an elementary Σ -strategy, σ is

- *winning*, if for every configuration $x \in \mathcal{C}^\infty(\sigma)$ that is *+maximal* (meaning that for every atomic extension $x \xrightarrow{a} \bar{c} \in \mathcal{C}^\infty(\sigma)$, $\text{pol}_A(a) = -$) then x is tautological.
- *top-winning* if $|\sigma| \in \mathcal{C}^\infty(\sigma)$ is tautological.

A winning Σ -strategy is top-winning, but not always the other way around. The *minimal, top-winning* Σ -strategies $\sigma : \llbracket \varphi \rrbracket^\exists$ will correspond to expansion trees; but the *winning strategies* will behave better compositionally.

5.3.2 A (biased) interpretation of formulas

We now complete the \exists -biased interpretation of formulas as games introduced in the previous section. This is performed inductively on the structure of formulas so we have to consider formulas that are not closed, *i.e.* with free variables.

\mathcal{V} -games For \mathcal{V} a finite set, a \mathcal{V} -game is defined as a wingame \mathcal{A} from Definition 5.7, but with signature Σ extended with \mathcal{V} . In other words, for $x \in \mathcal{C}^\infty(A)$,

$$\mathcal{W}_{\mathcal{A}}(x) \in \text{QF}_{\Sigma \uplus \mathcal{V}}^\infty(x).$$

We now define constructions on \mathcal{V} -games rather than just wingames. The duality operation on games extends to \mathcal{V} -games, simply by negating the winning conditions:

Definition 5.10. Let \mathcal{A} be a \mathcal{V} -game, its *dual* \mathcal{A}^\perp is the game A^\perp and for every $x \in \mathcal{C}^\infty(A)$ the winning condition,

$$\mathcal{W}_{\mathcal{A}^\perp}(x) = \mathcal{W}_{\mathcal{A}}(x)^\perp.$$

Similarly, the usual parallel composition of games gives rise to *two* constructions on \mathcal{V} -games:

Definition 5.11. Let \mathcal{A} and \mathcal{B} be \mathcal{V} -games. Their **tensor** $\mathcal{A} \otimes \mathcal{B}$ and their **par** $\mathcal{A} \wp \mathcal{B}$ have both $A \parallel B$ as underlying game, and winning conditions, for $x_A \parallel x_B \in \mathcal{C}^\infty(A \parallel B)$:

$$\begin{aligned} \mathcal{W}_{\mathcal{A} \otimes \mathcal{B}}(x_A \parallel x_B) &= \mathcal{W}_{\mathcal{A}}(x_A) \wedge \mathcal{W}_{\mathcal{B}}(x_B) \\ \mathcal{W}_{\mathcal{A} \wp \mathcal{B}}(x_A \parallel x_B) &= \mathcal{W}_{\mathcal{A}}(x_A) \vee \mathcal{W}_{\mathcal{B}}(x_B) \end{aligned}$$

There is an implicit renaming going on there: $\mathcal{W}_A(x_A), \mathcal{W}_B(x_B)$ are considered in $\mathbf{QF}_{\Sigma \uplus \mathcal{V}}^\infty(x_A \parallel x_B)$ rather than in $\mathbf{QF}_{\Sigma \uplus \mathcal{V}}^\infty(x_A)$ and $\mathbf{QF}_{\Sigma \uplus \mathcal{V}}^\infty(x_B)$ respectively – here and in the sequel, we will keep such renamings implicit

Note that in all three cases, we use the syntactic constructions $(-)^{\perp}$, \vee and \wedge on formulas. So the winning conditions generate uninterpreted syntactic objects. Note also that this definition yields the expected duality between the tensor and the par.

Lemma 5.1. *For any games \mathcal{A} and \mathcal{B} , we have:*

$$\begin{aligned} (\mathcal{A} \otimes \mathcal{B})^{\perp} &= \mathcal{A}^{\perp} \wp \mathcal{B}^{\perp} \\ (\mathcal{A} \wp \mathcal{B})^{\perp} &= \mathcal{A}^{\perp} \otimes \mathcal{B}^{\perp} \end{aligned}$$

Proof. Straightforward by definition and the De Morgan laws on formulas. \square

As expected, the interpretation of conjunction and disjunction relies on the above constructions:

$$\begin{aligned} \llbracket \varphi_1 \vee \varphi_2 \rrbracket_{\mathcal{V}}^{\exists} &= \llbracket \varphi_1 \rrbracket_{\mathcal{V}}^{\exists} \wp \llbracket \varphi_2 \rrbracket_{\mathcal{V}}^{\exists} \\ \llbracket \varphi_1 \wedge \varphi_2 \rrbracket_{\mathcal{V}}^{\exists} &= \llbracket \varphi_1 \rrbracket_{\mathcal{V}}^{\exists} \otimes \llbracket \varphi_2 \rrbracket_{\mathcal{V}}^{\exists} \end{aligned}$$

for $\varphi_1, \varphi_2 \in \mathbf{Form}(\mathcal{V})$ and the interpretation now parametrized by \mathcal{V} and producing \mathcal{V} -games.

The reader may wonder why these operations are written \otimes and \wp rather than \wedge and \vee . This is because, as we will see, these operations by themselves behave more like the connectives of linear logic [Gir87] than those of classical logic; for each \mathcal{V} the \otimes and \wp will form the basis of a $*$ -autonomous structure and hence a model of multiplicative linear logic.

Replication. To recover classical logic, we will add *replication* to the interpretation of formulas. We first define the countable parallel composition on games as follows.

Definition 5.12. Let A be an elementary game, its *countable parallel composition* is the game $\parallel_{\omega} A = (\mathbb{N} \times |A|, \leq_{\parallel_{\omega} A}, \text{pol}_{\parallel_{\omega} A})$ with

- causality: $(i, a_1) \leq_{\parallel_{\omega} A} (j, a_2)$ iff $i = j$ and $a_1 \leq_A a_2$;
- polarity: $\text{pol}_{\parallel_{\omega} A}((i, a)) = \text{pol}_A(a)$.

Configurations in $\parallel_{\omega} A$ are of the form $\parallel_{i \in \omega} x_i$ with $x_i \in \mathcal{C}^\infty(A)$. We say that $x \in \mathcal{C}^\infty(\parallel_{\omega} A)$ has *finite support* if it has only finitely many non-empty components – note that this is different from being finite. Again, this construction on games yields two distinct operation on \mathcal{V} -games:

Definition 5.13. Let \mathcal{A} be a \mathcal{V} -game. We define two new \mathcal{V} -games $!\mathcal{A}$ and $?\mathcal{A}$ with base-game $\parallel_{\omega} A$, and winning conditions:

$$\begin{aligned}\mathcal{W}_{!\mathcal{A}}(\parallel_{i \in \omega} x_i) &= \bigwedge_{i \in \omega} \mathcal{W}_{\mathcal{A}}(x_i) \\ \mathcal{W}_{?\mathcal{A}}(\parallel_{i \in \omega} x_i) &= \bigvee_{i \in \omega} \mathcal{W}_{\mathcal{A}}(x_i)\end{aligned}$$

Formally the above leads to infinite countable conjunctions or disjunctions even for finite configurations. When x has finite support though, we will always implicitly simplify it to a finite one, compacting the infinitely many occurrences of $\mathcal{W}_{\mathcal{A}}(\emptyset)$ into a single one.

Prefixing By the above, we know how to replicate moves in \mathcal{V} -games, we now define how to introduce new ones so as to interpret quantifiers.

Definition 5.14. Let A be an elementary game, its *+ -prefixed game* $+A$ has

- Events: $\{(0, \circ)\} \cup (\{1\} \times |A|)$;
- Causality: $(i, a) \leq_{+A} (j, a')$ iff $i = j = 1$ and $a \leq_A a'$, or $(i, a) = (0, \circ)$;
- Polarity: $\text{pol}_{+A}((0, \circ)) = +$ and $\text{pol}_{+A}((1, a)) = \text{pol}_A(a)$.

And similarly for the *- -prefixed game*, $-A$, with the exception of $\text{pol}_{-A}((0, \circ)) = -$.

Causality implies that that $(0, \circ)$ is the unique minimal event in $+A$ (respectively $-A$). Thus, configurations in prefixed game are either empty, or of the form $\{(0, \circ)\} \cup (\{1\} \times x_A)$ with $x_A \in \mathcal{C}^{\infty}(A)$, written $\circ.x_A$. When clear in the context we will drop the 0/1 indexing on events. Drawing inspiration from the game metaphor on expansion trees, we introduce two new constructions on \mathcal{V} -games:

Definition 5.15. For \mathcal{A} a $(\mathcal{V} \uplus \{x\})$ -game, the \mathcal{V} -game $\forall x.\mathcal{A}$ and its dual $\exists x.\mathcal{A}$ have games $-A$ and $+A$ respectively, and winning conditions:

$$\begin{aligned}\mathcal{W}_{\forall x.\mathcal{A}}(\emptyset) &= \top & \mathcal{W}_{\forall x.\mathcal{A}}(\forall.x_A) &= \mathcal{W}_{\mathcal{A}}(x_A)[\forall/x] \\ \mathcal{W}_{\exists x.\mathcal{A}}(\emptyset) &= \perp & \mathcal{W}_{\exists x.\mathcal{A}}(\exists.x_A) &= \mathcal{W}_{\mathcal{A}}(x_A)[\exists/x]\end{aligned}$$

abusing notation by writing \forall and \exists instead of $(0, \circ)$ for the initial move.

In other words, on a universal quantifier, Opponent is supposed to start; if he does not, that is a win for Player regardless of the rest of \mathcal{A} . Dually, on an existential quantifier Player is supposed to provide a witness and he

$$\begin{array}{ll}
\llbracket \top \rrbracket_{\mathcal{V}}^{\exists} = 1 & \llbracket \mathbf{P}(t_1, \dots, t_n) \rrbracket_{\mathcal{V}}^{\exists} = \mathbf{P}(t_1, \dots, t_n) \\
\llbracket \perp \rrbracket_{\mathcal{V}}^{\exists} = \perp & \llbracket \neg \mathbf{P}(t_1, \dots, t_n) \rrbracket_{\mathcal{V}}^{\exists} = \neg \mathbf{P}(t_1, \dots, t_n) \\
\llbracket \exists x \varphi \rrbracket_{\mathcal{V}}^{\exists} = ? \exists x. \llbracket \varphi \rrbracket_{\mathcal{V}_{\psi\{x\}}}^{\exists} & \llbracket \varphi_1 \vee \varphi_2 \rrbracket_{\mathcal{V}}^{\exists} = \llbracket \varphi_1 \rrbracket_{\mathcal{V}}^{\exists} \wp \llbracket \varphi_2 \rrbracket_{\mathcal{V}}^{\exists} \\
\llbracket \forall x \varphi \rrbracket_{\mathcal{V}}^{\exists} = \forall x. \llbracket \varphi \rrbracket_{\mathcal{V}_{\psi\{x\}}}^{\exists} & \llbracket \varphi_1 \wedge \varphi_2 \rrbracket_{\mathcal{V}}^{\exists} = \llbracket \varphi_1 \rrbracket_{\mathcal{V}}^{\exists} \otimes \llbracket \varphi_2 \rrbracket_{\mathcal{V}}^{\exists}
\end{array}$$

Figure 5.10: \exists -biased interpretation of formulas

loses if he fails to do so. In both cases, once the initial move has been played, we continue on \mathcal{A} with the variable x replaced with the newly introduced witness.

Putting everything together, we give in Figure 5.10 the general definition of the \exists -biased interpretation of a formula $\varphi \in \mathbf{Form}_{\Sigma}(\mathcal{V})$ as a \mathcal{V} -game. In particular, one can note the difference between the case of existential and universal formulas, reflecting the bias towards Eloïse in the interpretation. The reader can check that this is indeed compatible with the examples given previously.

Towards compositional Herbrand's theorem We can now restate Herbrand's theorem in term of elementary concurrent games:

Theorem 5.9. *For any closed formula φ , $\models \varphi$ iff φ has a finite, top-winning Σ -strategy $\sigma : \llbracket \varphi \rrbracket^{\exists}$.*

Besides the game-theoretic language, the difference with expansion trees is superficial: on φ , expansion trees essentially coincide with the *minimal* top-winning Σ -strategies $\sigma : \llbracket \varphi \rrbracket^{\exists}$; the only noticeable difference between the two is that strategies carry an explicit witness for acyclicity.

From that perspective, theorem 5.9 above can be deduced via Miller's version of Herbrand's theorem (see 5.8) with expansion trees. This would however make its validity intrinsically rely on the admissibility of cut in the sequent calculus and leave Σ -strategies as static objects, alternative bureaucracy-free representations of cut-free proofs.

But unlike expansion trees, strategies can be *composed*. In the next two chapters we will see that the effort to change view point, from a syntactic construction to a (game) semantic one, pays off as our games model will allow us to give an alternative proof of Herbrand's theorem where witnesses are obtained truly *compositionally* from any sequent proof, without first eliminating cuts. In other words, expansion trees will come naturally from the interpretation of the classical sequent calculus, as presented on figure 5.1, in the above games model.

As a first glimpse, a proof π of an LK sequent $\vdash \varphi_1, \dots, \varphi_n$ will be interpreted as a winning Σ -strategy $\llbracket \pi \rrbracket : \llbracket \varphi_1 \rrbracket \wp \dots \wp \llbracket \varphi_n \rrbracket$. Then, in order to interpret the cut rule,

$$\text{CUT} \frac{\vdash \Gamma, \varphi \quad \vdash \varphi^\perp, \Delta}{\vdash \Gamma, \Delta}$$

we will use the *composition* $\llbracket \pi_2 \rrbracket \odot \llbracket \pi_1 \rrbracket$ between $\llbracket \pi_1 \rrbracket : \llbracket \Gamma \rrbracket \wp \llbracket \varphi \rrbracket$ and $\llbracket \pi_2 \rrbracket : \llbracket \varphi \rrbracket^\perp \wp \llbracket \Delta \rrbracket$ resulting in a strategy over $\llbracket \Gamma \rrbracket \wp \llbracket \Delta \rrbracket$.

We will first make precise the linear part of this model (without replication) in order to interpret first order MLL, then put replication back to fully interpret the sequent rules of classical first order logic. The impatient reader may jump to theorem 7.2 for the final version this compositional Herbrand's theorem.

A model for first order MLL

In this chapter we show how the model of elementary games and Σ -strategies, Σ -Det, presented in chapter 2 and 4, can be extended with winning and a fibred structure in order to interpret first order MLL (MLL_1).

Following these two steps, we first show that \mathcal{V} -games and winning elementary Σ -strategies, as introduced in the previous chapter, form an $*$ -autonomous category \mathcal{V} -Games, that therefore is a model of \mathcal{V} -MLL.

We then show that the \mathcal{V} -Games categories support substitution and can be organised into a substitution-indexed $*$ -autonomous category. This allows for the interpretation of first order quantifier in the model and we finally interpret MLL_1 , proving a lax soundness result with respect to cut elimination.

6.1 Winning Σ -strategies

In this section we present a $*$ -autonomous category of games and strategies, obtained by extending the compact closed category Σ -Det presented in chapter 4 with winning conditions. We briefly recall why $*$ -autonomous categories are models of MLL.

In this section we aim to give an interpretation of MLL proofs, which should be invariant under cut-elimination. Categorical logic tells us that this is essentially the same as producing a *$*$ -autonomous category*. We opt here for the equivalent formulation by Cockett and Seely as a *symmetric linearly distributive category with negation* [CS97].

Definition 6.1. A *symmetric linearly distributive category* is a category \mathcal{C} with two symmetric monoidal structures $(\otimes, 1)$ and (\wp, \perp) which *distribute*: there is a natural

$$\delta_{A,B,C} : A \otimes (B \wp C) \xrightarrow{\mathcal{C}} (A \otimes B) \wp C,$$

the *linear distribution*, subject to the expected coherence conditions [CS97].

Definition 6.2. A symmetric linearly distributive category *with negation* also has a function $(-)^{\perp}$ on objects and families of maps

$$\eta_A : 1 \xrightarrow{\mathcal{C}} A^{\perp} \wp A \quad \text{and} \quad \epsilon_A : A \otimes A^{\perp} \xrightarrow{\mathcal{C}} \perp$$

such that the canonical composition $A \rightarrow A \otimes (A^\perp \wp A) \rightarrow (A \otimes A^\perp) \wp A \rightarrow A$, and its dual $A^\perp \rightarrow A^\perp$, are identities.

It is interesting to note that a *compact closed category* is a degenerate case of symmetric linearly distributive category where the monoidal structures $(\otimes, 1)$ and (\wp, \perp) coincide.

Abusing terminology, we refer to *symmetric linearly distributive categories with negation* by the shorter **-autonomous categories*. This should not create any confusion in the light of their equivalence.

6.1.1 *-autonomous structure

We now prove that the compact closed category of elementary games and Σ -strategies, Σ -Det as defined in section 4.3.3, can be turned into a symmetric linearly distributive category with negation – or equivalently, a *-autonomous category – by the addition of winning conditions onto games as presented in chapter 5.3.2.

In the following, we will use the wording Σ -receptivity and Σ -courtesy in place of \mathbb{T} -receptivity and \mathbb{T} -courtesy in the definition of elementary Σ -strategies (see definition 4.28).

A category of winning Σ -strategies

In section 5.3.2 of the previous chapter, we defined a notion of winning for elementary Σ -strategies over wingames (see definition 5.9). From now on we will restrict to those elementary Σ -strategies that are winning and simply refer to them as *winning strategies*.

On wingames, parallel composition is not defined anymore; instead there are two distinct constructors, \otimes and \wp , for putting games in parallel. We thus need to refine the usual definition of strategy from a game to an other:

Definition 6.3. Let \mathcal{A} and \mathcal{B} be wingames. A *winning strategy from \mathcal{A} to \mathcal{B}* is a winning elementary Σ -strategy $\sigma : \mathcal{A}^\perp \wp \mathcal{B}$.

We also write $\sigma : \mathcal{A} \xrightarrow{\text{Games}} \mathcal{B}$.

First of all, we give sufficient conditions for copycat strategies to be winning.

Lemma 6.1. *Let \mathcal{A}, \mathcal{B} be two wingames, and $f : A \cong B$ be an isomorphism of elementary games. Assume moreover that f preserves winning, in the sense that for all $x \in \mathcal{C}(A)$, we have*

$$\models \mathcal{W}_{\mathcal{A}}(x) \implies \mathcal{W}_{\mathcal{B}}(f x)[f^{-1}]$$

(we then write $f : \mathcal{A} \rightarrow \mathcal{B}$). Then, $\alpha_f : \mathcal{A} \rightarrow \mathcal{B}$ is a winning strategy from \mathcal{A} to \mathcal{B} .

Proof. Recall if $x_A \parallel x_B \in \mathcal{C}^\infty(\alpha_f)$ then $x_B = f x'_A$ such that $x'_A \in \mathcal{C}^\infty(A)$ and $x'_A \sqsubseteq x_A$. If, furthermore, $x_A \parallel x_B$ is +-maximal then, in fact, $x_A = x'_A$. Indeed, if there was $f a \in x_B$ with $a \notin x_A$, then $(x_A \cup \{a\}) \parallel x_B$ would be a positive extension of $x_A \parallel x_B$ still in $\mathcal{C}(\alpha_f)$, and similarly for the converse inclusion.

By definition, we have:

$$\mathcal{W}_{\mathcal{A} \rightarrow \mathcal{B}}(x_A \parallel f x_A) = \mathcal{W}_{\mathcal{A}}(x_A) \implies \mathcal{W}_{\mathcal{B}}(f x_A)$$

To check that $x_A \parallel x_B$ is tautological, we need to check that the substitution of the above by λ_{α_f} yields a tautology. Recall that λ_{α_f} leaves negative events unchanged, and replaces positive events with their negative counterpart on the other side. Hence, we have:

$$\begin{aligned} & \mathcal{W}_{\mathcal{A} \rightarrow \mathcal{B}}(x_A \parallel f x_A)[\lambda_{\alpha_f}] \\ &= \mathcal{W}_{\mathcal{A}}(x_A)[f a/a \mid a^- \in A] \implies \mathcal{W}_{\mathcal{B}}(f x_A)[a/f a \mid a^+ \in A] \end{aligned}$$

Applying the global renaming exchanging a and $f a$ for $a^- \in A$ (which preserves and reflects tautological status), we get:

$$\mathcal{W}_{\mathcal{A}}(x_A) \implies \mathcal{W}_{\mathcal{B}}(f x_A)[f^{-1}]$$

a tautology by assumption – in fact, we have proved that preserving winning in the sense above is a necessary and sufficient so that the corresponding copycat strategy is winning, though we shall only use the sufficient part in the sequel. \square

In particular, it follows from this lemma that for all wingame \mathcal{A} , $\alpha_{\mathcal{A}} : \mathcal{A} \rightarrow \mathcal{A}$ is a winning strategy. We now prove that winning strategies are stable under composition.

Lemma 6.2. *Let $\sigma : \mathcal{A} \rightarrow \mathcal{B}$ and $\tau : \mathcal{B} \rightarrow \mathcal{C}$ be two winning strategies. Then, $\tau \circ \sigma : \mathcal{A} \rightarrow \mathcal{C}$ is still a winning strategy.*

Proof. Let $x_A \parallel x_C \in \mathcal{C}^\infty(\tau \circ \sigma)$ be a +-maximal configuration, and consider its minimal witness $[x_A \parallel x_C]_{\tau \circ \sigma} \in \mathcal{C}^\infty(\tau \circ \sigma)$. We know that it is of the form

$$x_A \parallel x_B \parallel x_C$$

with $x_A \parallel x_B \in \mathcal{C}^\infty(\sigma)$ and $x_B \parallel x_C \in \mathcal{C}^\infty(\tau)$. Unfortunately, these two configurations might not be +-maximal in their respective strategies. Yet, consider the set of all witnesses for $x_A \parallel x_C \in \mathcal{C}^\infty(\tau \odot \sigma)$, *i.e.* the set

$$\{x_A \parallel x'_B \parallel x_C \in \mathcal{C}^\infty(\tau \circledast \sigma)\}$$

partially ordered by inclusion. It has suprema of all chains – as it is stable under unions – therefore, by Zorn's lemma, it has a maximal element:

$$x_A \parallel x_B^{\max} \parallel x_C \in \mathcal{C}^\infty(\tau \circledast \sigma)$$

This time, $x_A \parallel x_B^{\max} \in \mathcal{C}^\infty(\sigma)$ and $x_B^{\max} \parallel x_C \in \mathcal{C}^\infty(\tau)$ are +-maximal, for if it was not the case, we would get a contradiction with either maximality of $(x_A \parallel x_B^{\max} \parallel x_C)$ or +-maximality of $x_A \parallel x_C \in \mathcal{C}^\infty(\tau \odot \sigma)$.

From the fact that σ and τ are winning, we get then:

$$\begin{aligned} & \models (\mathcal{W}_A(x_A) \implies \mathcal{W}_B(x_B^{\max}))[\lambda_\sigma] \\ & \models (\mathcal{W}_B(x_B^{\max}) \implies \mathcal{W}_C(x_C))[\lambda_\tau] \end{aligned}$$

Since tautologies are stable under substitution, it follows:

$$\begin{aligned} & \models (\mathcal{W}_A(x_A) \implies \mathcal{W}_B(x_B^{\max}))[\lambda_\sigma][\lambda_{\tau \circledast \sigma}] \\ & \models (\mathcal{W}_B(x_B^{\max}) \implies \mathcal{W}_C(x_C))[\lambda_\tau][\lambda_{\tau \circledast \sigma}] \end{aligned}$$

(where there is, again, an implicit renaming of the free variables so that the expression typechecks). However, following the inductive definition of $\lambda_{\tau \circledast \sigma}$ (see 4.29) and the fact that λ_σ and λ_τ are idempotent, this is equivalent to

$$\begin{aligned} & \models (\mathcal{W}_A(x_A) \implies \mathcal{W}_B(x_B^{\max}))[\lambda_{\tau \circledast \sigma}] \\ & \models (\mathcal{W}_B(x_B^{\max}) \implies \mathcal{W}_C(x_C))[\lambda_{\tau \circledast \sigma}] \end{aligned}$$

therefore, by transitivity of implication,

$$\models (\mathcal{W}_A(x_A) \implies \mathcal{W}_C(x_C))[\lambda_{\tau \circledast \sigma}]$$

but this is the same as $(\mathcal{W}_A(x_A) \implies \mathcal{W}_C(x_C))[\lambda_{\tau \odot \sigma}]$, and hence $\tau \odot \sigma$ is as required a winning strategy. \square

Overall, we have proved:

Corollary 6.1. *There is a category Games of wingames and winning strategies.*

Bifunctors for \otimes and \wp

We now prove that the split of the parallel composition of games, into the tensor and par operations on wingames, as defined in section 5.3.2, extends to winning strategies and yields bifunctorial actions for both tensor and par.

Lemma 6.3. *For any $\sigma : \mathcal{A}_1 \dashv\vdash \mathcal{B}_1$ and $\tau : \mathcal{A}_2 \dashv\vdash \mathcal{B}_2$, the elementary Σ -strategy $\sigma \otimes \tau : \mathcal{A}_1 \otimes \mathcal{A}_2 \xrightarrow{\Sigma\text{-Det}} \mathcal{B}_1 \otimes \mathcal{B}_2$ defines two winning strategies:*

$$\sigma \otimes \tau : \mathcal{A}_1 \otimes \mathcal{A}_2 \dashv\vdash \mathcal{B}_1 \otimes \mathcal{B}_2 \quad ; \quad \sigma \wp \tau : \mathcal{A}_1 \wp \mathcal{A}_2 \dashv\vdash \mathcal{B}_1 \wp \mathcal{B}_2.$$

Proof. Let $(x_{A_1} \parallel x_{A_2}) \parallel (x_{B_1} \parallel x_{B_2}) \in \mathcal{C}^\infty(\sigma \otimes \tau)$ be $+$ -maximal. By definition of $\sigma \otimes \tau$, we have

$$(x_{A_1} \parallel x_{B_1}) \parallel (x_{A_2} \parallel x_{B_2}) \in \mathcal{C}^\infty(\sigma \parallel \tau)$$

which is $+$ -maximal as well, *i.e.* $x_{A_1} \parallel x_{B_1} \in \mathcal{C}^\infty(\sigma)$ and $x_{A_2} \parallel x_{B_2} \in \mathcal{C}^\infty(\tau)$ are $+$ -maximal. Since σ and τ are winning, it follows that:

$$\begin{aligned} &\models (\mathcal{W}_{\mathcal{A}_1}(x_{A_1}) \implies \mathcal{W}_{\mathcal{B}_1}(x_{B_1}))[\lambda_\sigma] \\ &\models (\mathcal{W}_{\mathcal{A}_2}(x_{A_2}) \implies \mathcal{W}_{\mathcal{B}_2}(x_{B_2}))[\lambda_\tau] \end{aligned}$$

Therefore, we have:

$$\models (\mathcal{W}_{\mathcal{A}_1}(x_{A_1})[\lambda_\sigma] \wedge \mathcal{W}_{\mathcal{A}_2}(x_{A_2})[\lambda_\tau]) \implies (\mathcal{W}_{\mathcal{B}_1}(x_{B_1})[\lambda_\sigma] \wedge \mathcal{W}_{\mathcal{B}_2}(x_{B_2})[\lambda_\tau])$$

by monotonicity of \wedge . But that is the same as:

$$\models ((\mathcal{W}_{\mathcal{A}_1}(x_{A_1}) \wedge \mathcal{W}_{\mathcal{A}_2}(x_{A_2})) \implies (\mathcal{W}_{\mathcal{B}_1}(x_{B_1}) \wedge \mathcal{W}_{\mathcal{B}_2}(x_{B_2})))[\lambda_{\sigma \otimes \tau}]$$

(leaving as usual some renamings implicit) as required.

Likewise, for $\sigma \wp \tau$, critically using monotonicity of \vee instead. \square

Since we already know that these operations preserve identities, we have finished constructing the bifunctorial action of \otimes and \wp :

$$\begin{aligned} - \otimes - & : \text{Games} \times \text{Games} \rightarrow \text{Games} \\ - \wp - & : \text{Games} \times \text{Games} \rightarrow \text{Games} \end{aligned}$$

A symmetric linearly distributive category

We now equip the category Games, together with the two bifunctors \otimes and \wp , with the structure of a symmetric linearly distributive category. Unsurprisingly, their units match with the interpretation of \top and \perp presented in section 5.3.

Definition 6.4. The *wingames* 1 and \perp are set to be the empty game \emptyset together with the respective winning conditions:

$$\mathcal{W}_1(\emptyset) = \top \quad \mathcal{W}_\perp(\emptyset) = \perp$$

With that in place, we state and prove the main result of this subsection.

Proposition 6.1. *The category Games is a symmetric linearly distributive category.*

Proof. We first check that $(\otimes, \mathbf{1})$ and (\wp, \perp) define symmetric monoidal structures on Games. All necessary structural strategies are copycat strategies, obtained using Lemma 6.1 and the observation that the following canonical isomorphisms of elementary games all preserve winning, and so do their inverse.

$$\begin{array}{llll} \rho_{\mathcal{A}}^{\otimes} : \mathcal{A} \otimes 1 & \rightarrow \mathcal{A} & \rho_{\mathcal{A}}^{\wp} : \mathcal{A} \wp \perp & \rightarrow \mathcal{A} \\ \lambda_{\mathcal{A}}^{\otimes} : 1 \otimes \mathcal{A} & \rightarrow \mathcal{A} & \lambda_{\mathcal{A}}^{\wp} : \perp \wp \mathcal{A} & \rightarrow \mathcal{A} \\ s_{\mathcal{A}, \mathcal{B}}^{\otimes} : \mathcal{A} \otimes \mathcal{B} & \rightarrow \mathcal{B} \otimes \mathcal{A} & s_{\mathcal{A}, \mathcal{B}}^{\wp} : \mathcal{A} \wp \mathcal{B} & \rightarrow \mathcal{B} \wp \mathcal{A} \\ \alpha_{\mathcal{A}, \mathcal{B}, \mathcal{C}}^{\otimes} : (\mathcal{A} \otimes \mathcal{B}) \otimes \mathcal{C} & \rightarrow \mathcal{A} \otimes (\mathcal{B} \otimes \mathcal{C}) & \alpha_{\mathcal{A}, \mathcal{B}, \mathcal{C}}^{\wp} : (\mathcal{A} \wp \mathcal{B}) \wp \mathcal{C} & \rightarrow \mathcal{A} \wp (\mathcal{B} \wp \mathcal{C}) \end{array}$$

This is immediate as they correspond to equivalences at the propositional level.

Then, there is a faithful forgetful functor from Games to Σ -Det sending both 1 and \perp to 1 , both \otimes and \wp to \otimes in the strict sense; and each of the copycat winning strategies above to the corresponding structural morphism for the symmetric monoidal structure of Σ -Det. It follows automatically that they satisfy the required coherence and naturality conditions, equipping Games with two symmetric monoidal structures $(\text{Games}, \otimes, \mathbf{1})$ and $(\text{Games}, \wp, \perp)$.

Finally, we check that there is a *linear distributivity* natural transformation. We notice that the associativity isomorphism of games $\alpha_{\mathcal{A}, \mathcal{B}, \mathcal{C}}$ also preserves winning:

$$\alpha_{\mathcal{A}, \mathcal{B}, \mathcal{C}} : \mathcal{A} \otimes (\mathcal{B} \wp \mathcal{C}) \rightarrow (\mathcal{A} \otimes \mathcal{B}) \wp \mathcal{C}$$

which boils down to the fact that for any $\varphi_1, \varphi_2, \varphi_3 \in \text{QF}_{\Sigma}^{\infty}(\mathcal{V})$,

$$\models \varphi_1 \wedge (\varphi_2 \vee \varphi_3) \implies (\varphi_1 \wedge \varphi_2) \vee \varphi_3$$

Note that unlike previously, the inverse of this isomorphism of games does not preserve winning. The coherence [CS97] and naturality conditions are again direct through the strict monoidal faithful forgetful operation, as their image in Σ -Det correspond to commuting diagrams of structural morphisms. \square

Negation

Our last point is to show that the symmetric linearly distributive category Games has negation. Again, we will do that by showing that the units and co-units for the compact closed structure of Σ -Det can be enriched with winning conditions.

Lemma 6.4. *Let \mathcal{A} be any wingame. Then, the following are winning strategies:*

$$\eta_{\mathcal{A}} : 1 \multimap \mathcal{A}^\perp \wp \mathcal{A} \quad \epsilon_{\mathcal{A}} : \mathcal{A} \otimes \mathcal{A}^\perp \multimap \perp$$

Proof. The proof follows the exact same lines as for Lemma 6.1. By a direct analysis of $+$ -maximal configurations and their annotations, the lemma boils down to the fact that for any formula φ , the following are tautologies.

$$\begin{aligned} \models \top &\implies \varphi^\perp \vee \varphi \\ \models \varphi \wedge \varphi^\perp &\implies \perp \end{aligned}$$

Through the same faithful forgetful operation as above, the required equations follow from the corresponding equations for the compact closure of Σ -Det. \square

Together, we have finished the proof of:

Corollary 6.2. *The category Games is a $*$ -autonomous category.*

6.1.2 Interpretation of MLL

It is known that if \mathcal{C} is a $*$ -autonomous category equipped with a choice of $\llbracket \mathbb{P}(t_1, \dots, t_n) \rrbracket$ (an object of \mathcal{C}) for all closed literals, then it defines a model for MLL based on those literals [See87]. Based on our game model Games, we give a brief overview of this standard interpretation of MLL proofs into a $*$ -autonomous category.

Formulas Following the \exists -biased interpretation of section 5.3, we regard a closed literal φ as the wingame $(\emptyset, \mathcal{W}_\varphi(\emptyset) = \varphi)$. For all such φ , we have $\llbracket \neg\varphi \rrbracket = \llbracket \varphi \rrbracket^\perp$.

By induction on the formulas, this interpretation then extends to all MLL formulas (or, alternatively, to all closed quantifier-free formulas) following the $*$ -autonomous structure of Games as depicted on figure 6.1.

$$\begin{array}{ll}
\llbracket \top \rrbracket = 1 & \llbracket \varphi_1 \vee \varphi_2 \rrbracket = \llbracket \varphi_1 \rrbracket \wp \llbracket \varphi_2 \rrbracket \\
\llbracket \perp \rrbracket = \perp & \llbracket \varphi_1 \wedge \varphi_2 \rrbracket = \llbracket \varphi_1 \rrbracket \otimes \llbracket \varphi_2 \rrbracket \\
\llbracket \neg \varphi \rrbracket = \llbracket \varphi \rrbracket^\perp & \llbracket P(t_1, \dots, t_n) \rrbracket = P(t_1, \dots, t_n)
\end{array}$$

Figure 6.1: Interpretation of closed quantifier-free formulas of MLL_1

Proofs. A proof π of an MLL sequent $\vdash \varphi_1, \dots, \varphi_n$ is interpreted as a winning strategy

$$\llbracket \pi \rrbracket : 1 \xrightarrow{\text{Games}} \llbracket \varphi_1 \rrbracket \wp \dots \wp \llbracket \varphi_n \rrbracket.$$

This follows the standard lines of the interpretation of MLL into a $*$ -autonomous category as detailed on figure 6.2. By definition, this yields, for any proof, a winning strategy in the corresponding game. Furthermore, the cut rule being interpreted using composition, the categorical laws make this interpretation invariant under cut reduction. We have:

Theorem 6.1. *If $\pi \rightsquigarrow_{MLL} \pi'$ are proofs of $\vdash \Gamma$ then $\llbracket \pi \rrbracket = \llbracket \pi' \rrbracket$.*

So a proof has the same denotation as its cut-free form obtained by theorem 5.3. Note that in our case, the above theorem is trivial: every MLL formula φ is interpreted as an empty wingame of winning condition $w_{\llbracket \varphi \rrbracket}(\emptyset) = \varphi$. Hence, if φ is valid, all its proofs are sent to the empty strategy, that is winning by definition. So proofs are equalised in their interpretations by the mere fact of proving the same formulas and none of the structure of the proofs is yet reflected in our strategies. The disappointed reader can however be reassured, this interpretation is only a base step toward the richer interpretation of MLL_1 described in the next section.

6.2 Interpretation of MLL_1

As mentioned in section 5.3, interpreting first order logic starts by being able to interpret open formulas. As then intuited, this is achieved in our game model by allowing winning conditions to depend on free variables from a fresh finite set \mathcal{V} . This generalisation extends to strategies, simply allowing their term annotations to rely on \mathcal{V} as well.

Definition 6.5. Let \mathcal{A} and \mathcal{B} be two \mathcal{V} -games. A *winning \mathcal{V} -strategy from \mathcal{A} to \mathcal{B}* is a winning elementary $\{\Sigma \uplus \mathcal{V}\}$ -strategy $\sigma : \mathcal{A}^\perp \wp \mathcal{B}$.

We also write $\sigma : \mathcal{A} \xrightarrow{\text{Ga}_{\Sigma \uplus \mathcal{V}}} \mathcal{B}$.

It is clear that the $*$ -autonomous structure of Games is preserved by this generalisation; we note \mathcal{V} -Games the corresponding $*$ -autonomous category.

$$\begin{aligned}
\left[\left[\text{Ax} \frac{}{\vdash^{\mathcal{V}} \varphi^{\perp}, \varphi} \right] \right] &= 1 \xrightarrow{\eta_{\varphi}} \varphi^{\perp} \wp \varphi & \left[\left[\text{1I} \frac{}{\vdash^{\mathcal{V}} 1} \right] \right] &= 1 \rightarrow 1 \\
\left[\left[\text{⊥I} \frac{\frac{\pi}{\vdash^{\mathcal{V}} \Gamma}}{\vdash^{\mathcal{V}} \Gamma, \perp} \right] \right] &= 1 \xrightarrow{[\pi]} \Gamma \cong \Gamma \wp \perp \\
\left[\left[\text{Ex} \frac{\frac{\pi}{\vdash^{\mathcal{V}} \Gamma, \varphi, \psi, \Delta}}{\vdash^{\mathcal{V}} \Gamma, \psi, \varphi, \Delta} \right] \right] &= 1 \xrightarrow{[\pi]} \Gamma \wp \varphi \wp \psi \wp \Delta \cong \Gamma \wp \psi \wp \varphi \wp \Delta \\
\left[\left[\text{Cut} \frac{\frac{\frac{\pi_1}{\vdash^{\mathcal{V}} \Gamma, \varphi} \quad \frac{\pi_2}{\vdash^{\mathcal{V}} \varphi^{\perp}, \Delta}}{\vdash^{\mathcal{V}} \Gamma, \Delta}}{} \right] \right] &= 1 \xrightarrow{[\pi_1] \otimes [\pi_2]} \\
(\Gamma \wp \varphi) \otimes (\varphi^{\perp} \wp \Delta) &\rightarrow \Gamma \wp (\varphi \otimes \varphi^{\perp}) \wp \Delta \xrightarrow{\Gamma \otimes \epsilon_{\varphi} \otimes \Delta} \Gamma \wp \perp \wp \Delta \cong \Gamma \wp \Delta \\
\left[\left[\otimes \text{I} \frac{\frac{\frac{\pi_1}{\vdash^{\mathcal{V}} \Gamma, \varphi} \quad \frac{\pi_2}{\vdash^{\mathcal{V}} \psi, \Delta}}{\vdash^{\mathcal{V}} \Gamma, \varphi \otimes \psi, \Delta}}{} \right] \right] &= 1 \xrightarrow{[\pi_1] \otimes [\pi_2]} (\Gamma \wp \varphi) \otimes (\psi \wp \Delta) \rightarrow \Gamma \wp (\varphi \otimes \psi) \wp \Delta \\
\left[\left[\wp \text{I} \frac{\frac{\pi}{\vdash^{\mathcal{V}} \Gamma, \varphi, \psi, \Delta}}{\vdash^{\mathcal{V}} \Gamma, \varphi \wp \psi, \Delta} \right] \right] &= 1 \xrightarrow{[\pi]} \Gamma \wp \varphi \wp \psi \wp \Delta \cong \Gamma \wp (\varphi \wp \psi) \wp \Delta
\end{aligned}$$

Figure 6.2: Interpretation of MLL

$$\begin{array}{ll}
\llbracket 1 \rrbracket_{\mathcal{V}} = 1 & \llbracket \varphi_1 \wp \varphi_2 \rrbracket_{\mathcal{V}} = \llbracket \varphi_1 \rrbracket_{\mathcal{V}} \wp \llbracket \varphi_2 \rrbracket_{\mathcal{V}} \\
\llbracket \perp \rrbracket_{\mathcal{V}} = \perp & \llbracket \varphi_1 \otimes \varphi_2 \rrbracket_{\mathcal{V}} = \llbracket \varphi_1 \rrbracket_{\mathcal{V}} \otimes \llbracket \varphi_2 \rrbracket_{\mathcal{V}} \\
\llbracket \mathbf{P}(t_1, \dots, t_n) \rrbracket_{\mathcal{V}} = \mathbf{P}(t_1, \dots, t_n) & \llbracket \exists x \varphi \rrbracket_{\mathcal{V}} = \exists x. \llbracket \varphi \rrbracket_{\mathcal{V}\{x\}} \\
\llbracket \neg \mathbf{P}(t_1, \dots, t_n) \rrbracket_{\mathcal{V}} = \neg \mathbf{P}(t_1, \dots, t_n) & \llbracket \forall x \varphi \rrbracket_{\mathcal{V}} = \forall x. \llbracket \varphi \rrbracket_{\mathcal{V}\{x\}}
\end{array}$$

Figure 6.3: Interpretation of quantifier free formulas in \mathcal{V} -Games

Then the interpretation of MLL in Games as described in the previous section extends straightforwardly to \mathcal{V} -MLL in \mathcal{V} -Games, noted $\llbracket - \rrbracket_{\mathcal{V}}$; the only novelty is that \mathcal{V} -open literals φ are now allowed. Still their interpretation remains

$$\llbracket \varphi \rrbracket_{\mathcal{V}} = (\emptyset, \{\} \mapsto \varphi)$$

that is, the empty game with φ as winning condition on the unique (empty) configuration. Figure 6.3 summarizes the $\llbracket - \rrbracket_{\mathcal{V}}$ interpretation.

Of course, interpreting open formulas is not enough. In section 5.3 we also introduced two constructors on games, $\forall x$ and $\exists x$, to interpret quantifiers. Contrary to the biased interpretation $\llbracket - \rrbracket^{\exists}$ – where the replication constructor $?$ was needed for existentials – we show in this section that those constructors are well-suited to interpret directly the *linear* quantifiers of MLL₁. In particular, we give two constructions on proofs to interpret their introduction rules.

Checking that this interpretation is well-behaved for cuts however requires a bit of prior work. As mentioned in section 5.1, cut reduction rules for quantifiers make use of substitution on proofs (see figure 5.3). Our first objective is thus to introduce substitution on games and strategies in order to reflect this construction from proofs. This leads to organising \mathcal{V} -game models as an indexed category (yet to be defined) on top of the category of substitutions.

The informed reader may have recognised Lawvere’s category theoretic descriptions of quantifiers via *hyperdoctrine* [Law69], in the plan above. Although our work is certainly inspired from this notion, we will see that our model does not fit all of its properties.

6.2.1 A fibred model of \mathcal{V} -MLL

Following [Law69, See83], we expect to model \mathcal{V} -MLL and substitution on proofs in the categorical structure of an indexed category on top of the category of substitutions:

Definition 6.6. Let $\ast\text{-Aut}$ be the category of \ast -autonomous categories and functors preserving the structure strictly, then a *strict*¹ *Subst-indexed \ast -autonomous category* is a functor

$$\mathcal{G} : \text{Subst}^{\text{op}} \rightarrow \ast\text{-Aut}$$

The intuition is that \mathcal{G} selects a model of \mathcal{V} -MLL and functoriality provides a notion of substitution between these models (hence on the interpretation of proofs). For this to be the case, one further asks for \mathcal{G} to *support* Σ , meaning that for $\mathcal{V}_n = \{x_1, \dots, x_n\}$, every predicate symbol P of arity less than or equal to n is associated with an object of $\mathcal{G}(\mathcal{V}_n)$ – its *interpretation* $\llbracket P \rrbracket_{\mathcal{V}_n}$ – and that this association satisfies the following equality, for every $t_1, \dots, t_n \in \text{Tm}_{\Sigma}(\mathcal{V})$

$$\llbracket P(t_1, \dots, t_n) \rrbracket = (\llbracket P \rrbracket_{\mathcal{V}_n})[t_1/x_1, \dots, t_n/x_n]$$

where, for readability, the action of $\mathcal{G}(\gamma : \mathcal{V}_1 \xrightarrow{\Sigma} \mathcal{V}_2) : \mathcal{G}(\mathcal{V}_2) \rightarrow \mathcal{G}(\mathcal{V}_1)$ is written $(_)[\gamma]$.

Substitution on games Let us now introduce our concrete structure. For any finite \mathcal{V} , the *fibre* $\mathcal{G}(\mathcal{V})$ is going to be the category \mathcal{V} -Games described in the introduction. Characterising the functorial action of \mathcal{G} in our model thus amounts to defining substitution on \mathcal{V} -games and winning \mathcal{V} -strategies. We define it first on \mathcal{V} -games:

Definition 6.7. Let $\gamma : \mathcal{V}_1 \xrightarrow{\Sigma} \mathcal{V}_2$, and \mathcal{A} be a \mathcal{V}_2 -game. Then the substituted \mathcal{V}_1 -game $\mathcal{A}[\gamma]$ has unchanged game A , and winning conditions for $x \in \mathcal{C}^{\infty}(A)$:

$$\mathcal{W}_{\mathcal{A}[\gamma]}(x) = \mathcal{W}_{\mathcal{A}}(x)[\gamma] \in \text{QF}_{\Sigma \uplus \mathcal{V}_1}^{\infty}(x)$$

It is obvious from this definition that substitution commutes with all operations on \mathcal{V} -games, *i.e.* $(-)^{\perp}$, \otimes and \wp . Moreover, substitution on games supports Σ , as, for any predicate P of arity $n \leq |\mathcal{V}|$, and any terms $t_1, \dots, t_n \in \text{Tm}_{\Sigma}(\mathcal{V})$:

$$\begin{aligned} \llbracket P(t_1, \dots, t_n) \rrbracket_{\mathcal{V}} &= (\emptyset, \{\} \mapsto P(t_1, \dots, t_n)) \\ &= (\emptyset, \{\} \mapsto P(x_1 \dots x_n)[t_1/x_1, \dots, t_n/x_n]) \\ &= (\llbracket P \rrbracket_{\mathcal{V}_n})[t_1/x_1, \dots, t_n/x_n] \end{aligned}$$

¹ Usually definitions (*e.g.* [See83]) ask for strong functors instead of strict ones, *i.e.* equality up to isomorphism instead of strict equality. We opt here for this simpler version as our own model turns out to be strict.

Substitution on winning strategies We now move on to defining substitution on winning \mathcal{V} -strategies; we define it first on $(\Sigma \uplus \mathcal{V})$ -augmentations.

Definition 6.8. Let $(\mathfrak{q}, \lambda_{\mathfrak{q}}) \in (\Sigma \uplus \mathcal{V}_2)\text{-Aug}(A)$ for some game A and $\gamma : \mathcal{V}_1 \xrightarrow{\Sigma} \mathcal{V}_2$ be a substitution, then *substitution of \mathfrak{q} by γ* is:

$$\mathfrak{q}[\gamma] = (\mathfrak{q}, \lambda_{\mathfrak{q}}[\gamma])$$

This obviously defines a $(\Sigma \uplus \mathcal{V}_1)$ -augmentation over A .

Checking that substitution is actually well-behaved on winning strategies requires a little work; we first inspect the term annotations.

Lemma 6.5. *Let $\sigma : A$ be an elementary $(\Sigma \uplus \mathcal{V}_2)$ -strategy, and $\gamma : \mathcal{V}_1 \xrightarrow{\Sigma} \mathcal{V}_2$ be a substitution. Then, $\sigma[\gamma]$ is an elementary $(\Sigma \uplus \mathcal{V}_1)$ -strategy on A .*

Proof. We only need to check Σ -receptivity and Σ -courtesy. For Σ -receptivity, take $a^- \in |\sigma|$. We know that $\lambda_{\sigma}(a) = a$, so in particular without variables in \mathcal{V}_2 . Therefore, $\lambda_{\sigma[\gamma]}(a) = a$ as well as required.

For Σ -courtesy, take $a^+ \in |\sigma|$. We have $\lambda_{\sigma}(a) \in \text{Tm}_{\Sigma \uplus \mathcal{V}_2}([a]_{\sigma}^-) = \text{Tm}_{\Sigma}([a]_{\sigma}^- \uplus \mathcal{V}_2)$ by Σ -courtesy (we use, here, the implicit assumption that sets of free variables are always disjoint from the sets of events in a game – this can be easily ensured *w.l.o.g.*). Therefore, $\lambda_{\sigma}(a)[\gamma] \in \text{Tm}_{\Sigma}([a]_{\sigma}^- \uplus \mathcal{V}_1)$ as required. \square

We now check that substitution also preserves winning:

Lemma 6.6. *Let $\gamma : \mathcal{V}_1 \xrightarrow{\Sigma} \mathcal{V}_2$ be a substitution, \mathcal{A} a \mathcal{V}_2 -game, and $\sigma : \mathcal{A}$ a winning \mathcal{V}_2 -strategy. Then,*

$$\sigma[\gamma] : \mathcal{A}[\gamma]$$

is a winning \mathcal{V}_1 -strategy.

Proof. Let $x \in \mathcal{C}^{\infty}(\sigma)$ be \exists -maximal. Since σ is winning, we have $\models \mathcal{W}_{\mathcal{A}}(x)[\lambda_{\sigma}]$.

Tautologies are stable under substitution, so $\mathcal{W}_{\mathcal{A}}(x)[\lambda_{\sigma}][\gamma]$ is a tautology as well. But that is the same as $\mathcal{W}_{\mathcal{A}}(x)[\gamma][\lambda_{\sigma[\gamma]}}$ as substitution by γ does not create variables in x . Hence, $\sigma[\gamma]$ is winning as required. \square

Indexed category We wish to prove that for $\gamma : \mathcal{V}_1 \xrightarrow{\Sigma} \mathcal{V}_2$, the substitution defined above extends into a strict $*$ -autonomous functor from $\mathcal{V}_2\text{-Games}$ to $\mathcal{V}_1\text{-Games}$. More generally,

Theorem 6.2. *Games and winning strategies can be organized into an indexed $*$ -autonomous category, that is a functor*

$$\text{Games}(-) : \text{Subst}^{\text{op}} \rightarrow *\text{-Aut}$$

where $*\text{-Aut}$ is the category of $*$ -autonomous categories and strict $*$ -autonomous functors.

Proof. We set $\text{Games}(\mathcal{V}) = \mathcal{V}\text{-Games}$ and for $\gamma : \mathcal{V}_1 \xrightarrow{\Sigma} \mathcal{V}_2$ a substitution, $\text{Games}(\gamma)$ is defined on \mathcal{V}_2 -games and winning \mathcal{V}_2 -strategies as explained in definitions 6.7 and 6.8.

We have already noticed that substitution is strict with respect to the operations on \mathcal{V} -games so we focus on structural morphisms and operations on winning strategies.

Since substitution only acts over term annotations but preserves causal structure and winning at the level of strategies, every proof obligation boils down to a check on term annotations.

For structural morphisms, it is enough to note that copycat strategies do not make use of variables in \mathcal{V}_2 in their term annotations. So they are strictly preserved by substitution. For tensor product, it is direct to check that substitution commutes with the corresponding renaming.

Composition is maybe the most intricate case: take $\sigma : A^\perp \parallel B$ and $\tau : B^\perp \parallel C$ two winning \mathcal{V}_2 -strategies, then, by definition, $\tau[\gamma] \otimes \sigma[\gamma] = (\tau \otimes \sigma)[\gamma]$ at the level of plain augmentations and, following the inductive definition of their respective term annotations (see 4.23) it is easy to check that

$$\lambda_{(\tau \otimes \sigma)[\gamma]} = \lambda_{(\tau[\gamma]) \otimes (\sigma[\gamma])}$$

Hence, the two $\Sigma \uplus \mathcal{V}_1$ -augmentations are actually equal and the same holds after hiding. So substitution preserves composition.

Finally, it is obvious by definition 6.7 and 6.8 that the above assignment for $\text{Games}(-)$ is itself functorial, yielding an indexed structure. \square

6.2.2 Linear quantifiers

For any finite set \mathcal{V} , the construction above provides a way to interpret \mathcal{V} -MLL in \mathcal{V} -Games and also to model substitutions. We finally give the interpretation for first order quantifiers and their introduction rules $\forall\text{I}$ and $\exists\text{I}$. We call this interpretation *linear* and write $\llbracket - \rrbracket_{\mathcal{V}}^{\ell}$. On formulas, $\llbracket - \rrbracket_{\mathcal{V}}^{\ell}$ extends $\llbracket - \rrbracket_{\mathcal{V}}$, interpreting quantifiers with the quantifiers on games from definition 5.15:

$$\llbracket \exists x \varphi \rrbracket_{\mathcal{V}}^{\ell} = \exists x. \llbracket \varphi \rrbracket_{\mathcal{V} \uplus x}^{\ell} \quad \llbracket \forall x \varphi \rrbracket_{\mathcal{V}}^{\ell} = \forall x. \llbracket \varphi \rrbracket_{\mathcal{V} \uplus x}^{\ell}$$

Note that this is almost the same interpretation as $\llbracket - \rrbracket_{\mathcal{V}}^{\exists}$ except for $\llbracket \exists x \varphi \rrbracket_{\mathcal{V}}^{\ell}$.

Besides preserving the $*$ -autonomous structure, substitution also propagates through quantifiers on games, from which we have:

Lemma 6.7. *Let $\varphi \in \text{Form}_{\Sigma}(\mathcal{V}_2)$ and $\gamma : \mathcal{V}_1 \rightarrow \mathcal{V}_2$ a substitution, then $\llbracket \varphi[\gamma] \rrbracket_{\mathcal{V}_1}^{\ell} = \llbracket \varphi \rrbracket_{\mathcal{V}_2}^{\ell}[\gamma]$.*

Proof. That the constructors $\exists x$ and $\forall x$ commute with substitution $[\gamma]$ and $[\gamma \uplus (x \mapsto x)]$ is direct from their definitions. The lemma then follows by structural induction. \square

The above identity will be used implicitly from now on.

No hyperdoctrine Following the usual interpretation of quantifiers in hyperdoctrines [Law69], the way to proceed from a Subst-indexed $*$ -autonomous category \mathcal{G} is to find, for every finite set \mathcal{V} , two functors, $\forall_{\mathcal{V},x}, \exists_{\mathcal{V},x} : \mathcal{G}(\mathcal{V} \uplus \{x\}) \rightarrow \mathcal{G}(\mathcal{V})$, that respectively are right and left adjoints to the weakening functor $[w_{\mathcal{V},x}] = \mathcal{G}(w_{\mathcal{V},x}) : \mathcal{G}(\mathcal{V}) \rightarrow \mathcal{G}(\mathcal{V} \uplus \{x\})$ – for $w_{\mathcal{V},x} : \mathcal{V} \uplus \{x\} \xrightarrow{\Sigma} \mathcal{V}$ the weakening substitution:

$$\begin{array}{ccc} & \forall_{\mathcal{V},x} & \\ & \top & \\ \mathcal{G}(\mathcal{V} \uplus \{x\}) & \xleftarrow{[w_{\mathcal{V},x}]} & \mathcal{G}(\mathcal{V}) \\ & \top & \\ & \exists_{\mathcal{V},x} & \end{array}$$

This leads to interpreting the $\forall I$ and $\exists I$ rules as follows:

- for $\llbracket \pi \rrbracket_{\mathcal{V} \uplus \{x\}} : A[w_{\mathcal{V},x}] \xrightarrow{\mathcal{G}(\mathcal{V} \uplus \{x\})} B$ a proof interpretation in $\mathcal{G}(\mathcal{V} \uplus \{x\})$, the right adjunction $\forall_{\mathcal{V},x} \vdash [w_{\mathcal{V},x}]$ yields a morphism

$$A \xrightarrow{\mathcal{G}(\mathcal{V})} \forall_{\mathcal{V},x} B.$$

This morphism corresponds to the interpretation of π followed by $\forall I$ in $\mathcal{G}(\mathcal{V})$;

- for $\llbracket \pi \rrbracket_{\mathcal{V}} : A \xrightarrow{\mathcal{G}(\mathcal{V})} B[t/x]$ a proof interpretation in $\mathcal{G}(\mathcal{V})$ – with $[t/x]$ the substitution functor $\mathcal{G}([t/x]) : \mathcal{G}(\mathcal{V} \uplus \{x\}) \rightarrow \mathcal{G}(\mathcal{V})$ –, the co-unit of the left adjunction $w_{\mathcal{V},x} \vdash \exists_{\mathcal{V},x}$ defines a morphism $\eta_B : B \xrightarrow{\mathcal{G}(\mathcal{V} \uplus \{x\})} (\exists_{\mathcal{V},x}(B))[w_{\mathcal{V},x}]$ which can be transported back to $\mathcal{G}(\mathcal{V})$ via $[t/x]$, leading to

$$(\eta_B)[t/x] : B[t/x] \xrightarrow{\mathcal{G}(\mathcal{V})} (\exists_{\mathcal{V},x}(B)).$$

This morphism can thus be used in post-composition with $\llbracket \pi \rrbracket$ to define a morphism $A \xrightarrow{\mathcal{G}(\mathcal{V})} (\mathcal{G}[t/x]\exists_{[t/x]})B$, the interpretation of π followed by $\exists\text{I}$ in $\mathcal{G}(\mathcal{V})$.

In both case naturality ensures that this is preserved by cut elimination.

In our model, the constructors $\forall x.$ and $\exists x.$ on \mathcal{V} -games defined in 5.15, extend to functors $\forall_{\mathcal{V},x}, \exists_{\mathcal{V},x} : (\mathcal{V} \uplus \{x\})\text{-Games} \rightarrow \mathcal{V}\text{-Games}$: for $\sigma : \mathcal{A}^\perp \wp \mathcal{B}$ a winning $\mathcal{V} \uplus \{x\}$ -strategy,

$$\forall_{\mathcal{V},x}(\sigma) : (\forall x. \mathcal{A})^\perp \wp \forall x. \mathcal{B}$$

is the winning \mathcal{V} -strategy that first plays copycat on the initial \forall , then keeps playing as σ ; and similarly for $\exists_{\mathcal{V},x}(\sigma) : (\exists x. \mathcal{A})^\perp \wp \exists x. \mathcal{B}$.

We do not expend further on these functorial constructions as, unfortunately, they do not define the expected left and right adjoints for $w_{\mathcal{V},x}$. Yet, investigating on the desired natural bijections lead us to constructions in which to interpret the introduction rules for quantifiers, which we now present. Later, we will see that this interpretation only preserves cut-elimination in a weak sense; this is related to the above functors not defining adjunctions.

$\exists\text{I}$ and $\forall\text{I}$ interpretation We now interpret the $\forall\text{I}$ and $\exists\text{I}$ introduction rules. First, we give an elementary Σ -strategy introducing a witness t (it plays the role of the substituted co-unit described above).

Definition 6.9. Let A be a game and $t \in \text{Tm}_\Sigma(\mathcal{V})$, we define $\exists_A^t : A^\perp \parallel \exists.A$ to be the tuple $(|A^\perp \parallel \exists.A|, \leq_{\exists_A^t}, \lambda_{\exists_A^t})$ where $\leq_{\exists_A^t}$ includes $\leq_{\mathfrak{c}_A}$, plus dependencies

$$\{((1, \exists), (1, a)) \mid a \in A\} \uplus \{((1, \exists), (0, a)) \mid \exists a_0 \in A^-. a_0 \leq_A a\}$$

and term assignment that of \mathfrak{c}_A plus $\lambda_{\exists_A^t}((1, \exists)) = t$.

This obviously defines an elementary $(\Sigma \uplus \mathcal{V})$ -strategy. This strategy plays \exists annotated with t , then proceeds as copycat on A . We have:

Lemma 6.8. *Let A be a \mathcal{V} -game, and $t \in \text{Tm}_\Sigma(\mathcal{V})$. Then,*

$$\exists_A^t : \mathcal{A}^\perp[t/x] \wp \exists x \mathcal{A}$$

is a winning \mathcal{V} -strategy.

Proof. Let $x_A \parallel x_{\exists A} \in \mathcal{C}^\infty(\exists_A^t)$ be a +-maximal configuration. As $(1, \exists)$ is minimal in \exists_A^t , and by property of +-maximal configurations of copycat, x necessarily has the form $x_A \parallel (\{(1, \exists)\} \cup x_A)$ where $x_A \in \mathcal{C}^\infty(A)$. It follows by definition of $\lambda_{\exists_A^t}$ that:

$$\begin{aligned} \mathcal{W}_{\mathcal{A}^+ [t/x] \wp \exists x \mathcal{A}}(x) [\lambda_{\exists_A^t}] &= (\mathcal{W}_A(x_A) [t/x]^\perp \vee \mathcal{W}_A(x_A) [\exists/x]) [t/\exists] \\ &= (\mathcal{W}_A(x_A)^\perp \vee \mathcal{W}_A(x_A)) [t/x] \end{aligned}$$

which is a tautology. \square

As suggested above, \exists_A^t serves in the semantic construction corresponding to the introduction rule for the existential quantifier: we interpret $\exists \mathbb{I}$ by post-composing with \exists_A^t . This can be depicted as – removing interpretation brackets around games for more readability:

$$\left[\exists \mathbb{I} \frac{\frac{\pi}{\vdash^{\mathcal{V}} \Gamma, \varphi [t/x]}}{\vdash^{\mathcal{V}} \Gamma, \exists x. \varphi} \right]^\ell = \Gamma^\perp \xrightarrow{[\pi]^\ell} \varphi [t/x] \xrightarrow{\exists_A^t} \exists x. \varphi$$

In the above, we wrote $[\pi]^\ell : \Gamma^\perp \dashv\vdash \varphi [t/x]$, silently using the natural isomorphism between $\mathbf{1} \dashv\vdash \Gamma \wp \varphi$ and $\Gamma^\perp \dashv\vdash \varphi$.

Symmetrically, we now introduce the semantic construction that will match introduction of the universal quantifier.

Definition 6.10. For σ an elementary $(\Sigma \uplus \mathcal{V} \uplus \{x\})$ -strategy on $A^\perp \parallel B$, we define $\forall_{A,B}^x(\sigma) : A^\perp \parallel \forall. B$ the elementary $(\Sigma \uplus \mathcal{V})$ -strategy having:

- Events: $|\sigma| \uplus \{(1, \forall)\}$;
- Causality: $\leq_\sigma \cup \{((1, \forall), s) \mid s \in \forall. B \vee \exists s' \leq_\sigma s, x \in \text{fv}(\lambda_\sigma(s'))\}$;
- Term assignment: $\lambda((1, \forall)) = (1, \forall)$ and $\lambda(s) = \lambda_\sigma(s)[(1, \forall)/x]$ ($s \in |\sigma|$).

Intuitively, $\forall_{A,B}^x(\sigma)$ waits for the new negative move before playing any event whose annotation includes x . However, it might still play positive moves on the left whose annotations do not include x . Additionally, in the term annotations, the variable x is changed to refer instead to the new negative event $(1, \forall)$.

We also prove the semantic correctness of the introduction of the universal quantifier (note its “adjointness” flavour) .

Proposition 6.2. *If σ is a winning $(\mathcal{V} \uplus \{x\})$ -strategy on $\mathcal{A}[\mathbf{w}_{\mathcal{V},x}] \wp \mathcal{B}$, then $\forall_{A,B}^x(\sigma)$ is winning on the \mathcal{V} -game $\mathcal{A} \wp \forall x. \mathcal{B}$.*

Proof. Let $x = x_A \parallel x_{\forall B} \in \mathcal{C}^\infty(\mathbb{M}_{A,B}^x(\sigma))$, be +-maximal, we distinguish two cases:

• If $(1, \forall) \notin x_{\forall B}$, then $x = x_A \parallel \emptyset$, and $\mathcal{W}_{\mathcal{A}^\perp \wp_{\forall x} \mathcal{B}}(x)[\lambda_{\mathbb{M}_{A,B}^x(\sigma)}] = \mathcal{W}_{\mathcal{A}}(x_A)[\lambda_{\mathbb{M}_{A,B}^x(\sigma)}]^\perp \vee \top$ is a tautology.

• Otherwise, $x = x_A \parallel \forall.x_B$ where $x_A \parallel x_B \in \mathcal{C}^\infty(\sigma)$ is +-maximal, and, using that \mathcal{A} is a \mathcal{V} -game (so x cannot appear in $\mathcal{W}_{\mathcal{A}}(x_A)$) we have:

$$\begin{aligned} \mathcal{W}_{\mathcal{A}^\perp \wp_{\forall x} \mathcal{B}}(x)[\lambda_{\mathbb{M}_{A,B}^x(\sigma)}] &= (\mathcal{W}_{\mathcal{A}}(x_A)^\perp \vee \mathcal{W}_{\mathcal{B}}(x_B)[(1, \forall)/x])[\lambda_\sigma][(1, \forall)/x] \\ &= (\mathcal{W}_{\mathcal{A}}(x_A)^\perp [(1, \forall)/x] \vee \mathcal{W}_{\mathcal{B}}(x_B)[(1, \forall)/x])[\lambda_\sigma][(1, \forall)/x] \\ &= (\mathcal{W}_{\mathcal{A}[w_{\forall, x}]}(x_A)^\perp \vee \mathcal{W}_{\mathcal{B}}(x_B))[\lambda_\sigma][(1, \forall)/x] \end{aligned}$$

a tautology, since σ is winning. \square

This completes the interpretation of MLL₁, setting

$$\left[\left[\frac{\pi}{\frac{\vdash^{\mathcal{V} \uplus \{x\}} \Gamma, \varphi}{\vdash^{\mathcal{V}} \Gamma, \forall x. \varphi}} \right] \right]^\ell = \Gamma^\perp \text{ (} \frac{\mathbb{M}(\llbracket \pi \rrbracket^\ell)}{\forall} \text{)} \forall x. \varphi$$

In the rest of this section, we study how the overall interpretation behaves with respect to cut-elimination. Associativity of composition already ensures that the interpretation of $\exists\mathbb{I}$ validates $\rightsquigarrow_{\text{CUT}/\exists}$, so we focus on $\rightsquigarrow_{\forall/\exists}$ and $\rightsquigarrow_{\text{CUT}/\forall}$.

Cancellation of introductions. As a first consistency check, we verify that the two introduction rules satisfy the equality corresponding to a cut between \forall and \exists , *i.e.* validates $\rightsquigarrow_{\forall/\exists}$.

Lemma 6.9. *Let $\sigma : \mathcal{A}^\perp \parallel \mathcal{B}$ be a winning $(\mathcal{V} \uplus \{x\})$ -strategy. Then, we have:*

$$(\exists_{\mathcal{B}}^t)^\perp \odot \forall I_{\mathcal{A}, \mathcal{B}}^x(\sigma) = \sigma[t/x]$$

where $(\exists_{\mathcal{B}}^x)^\perp$ is $\exists_{\mathcal{B}}^x$ viewed as a \mathcal{V} -strategy on $((\exists \mathcal{B})^\perp)^\perp \wp \mathcal{B}$.

Proof. The synchronization between $(1, \forall)$ and $(0, \exists)$ is minimal in the interaction, and causes a renaming of \mathbb{M}^x to t . After it is played, we get an interaction between σ and copycat on B .

More precisely let us show that $(\exists_{\mathcal{B}}^t)^\perp \odot \forall I_{\mathcal{A}, \mathcal{B}}^x(\sigma) = \mathfrak{c}_B \odot \sigma[t/x](= \sigma[t/x])$. Following proposition 4.3, we start by showing that these two elementary strategies – viewed as rigid ones – have the same set of augmentations.

By definition, an augmentation of $(\exists_B^t)^\perp$ is of the form $\emptyset \parallel x'_B$ or $(\exists x_B) \parallel x'_B$ such that $(\emptyset \parallel x'_B, \leq_{\exists_B^t})$, respectively $(x_B \parallel x'_B, \leq_{\exists_B^t})$, is a augmentation of \mathfrak{C}_B . Similarly an augmentation of $\forall_{\mathcal{A},\mathcal{B}}^x(\sigma)$ is of the form $x_A \parallel \emptyset$ or $x_A \parallel \forall^x . x_B$ such that $(x_A \parallel \emptyset, \leq_{\forall_{\mathcal{A},\mathcal{B}}^x(\sigma)})$, respectively $(x_A \parallel x_B, \leq_{\forall_{\mathcal{A},\mathcal{B}}^x(\sigma)})$ is an augmentation of σ . Note that in fact \exists and \forall denote the same prefixed event with reverse polarities, so, there is a mapping from the set of causally compatible augmentations of $(\exists_B^t)^\perp$ and $\forall_{\mathcal{A},\mathcal{B}}^x(\sigma)$, and the set of causally compatible augmentations of \mathfrak{C}_B and $\sigma[t/x]$. Moreover, if two pairs of compatible augmentations of $(\exists_B^t)^\perp$ and $\forall_{\mathcal{A},\mathcal{B}}^x(\sigma)$ are sent to the same pair of compatible augmentations of \mathfrak{C}_B and $\sigma[t/x]$ then, after interaction and projection, every pair define the same augmentation. Hence, by proposition 4.3 $(\exists_B^t)^\perp \odot \forall_{\mathcal{A},\mathcal{B}}^x(\sigma) = \mathfrak{C}_B \odot \sigma[t/x]$ as elementary strategies.

It remains to show that the term-annotations are equal: by definition $(\lambda_{\exists_B^t}) \upharpoonright_{B \parallel B} = \lambda_{\mathfrak{C}_B}$ and $(\lambda_{\forall_{\mathcal{A},\mathcal{B}}^x(\sigma)}) \upharpoonright_{A \parallel B} = \lambda_\sigma[\forall^x/x]$ so, following the inductive definition of term-annotation on interaction (see 4.29), it is easy to show that $(\lambda_{\exists_B^t \otimes \forall_{\mathcal{A},\mathcal{B}}^x(\sigma)}) \upharpoonright_{A \parallel B \parallel B} = \lambda_{\mathfrak{C}_A \otimes \sigma[t/x]}$ the key point being that $\lambda_{\exists_B^t \otimes \forall_{\mathcal{A},\mathcal{B}}^x(\sigma)}(\forall^x) = \lambda_{\exists_B^t}(\exists) = t$. \square

Weak commutation The above lemma proves that our interpretation leaves $\rightsquigarrow_{\forall/\exists}$ invariant; the counter-example below, however, shows that it fails $\rightsquigarrow_{\text{CUT}/\forall}$.

Example 6.1. Consider the following two first order classical proofs:

$$\frac{\frac{\frac{\overline{\vdash^{x,y} \perp, \top} \text{Ax}}{\vdash^{x,y} \exists x \perp, \top} \exists \text{I}, x := y}}{\vdash \exists z. \perp, \forall x \forall y. \top} \forall \text{I}}{\quad} \quad \frac{\frac{\frac{\overline{\vdash^z \perp, \top} \text{Ax}}{\vdash^z \exists x \exists y. \perp, \top} \exists \text{I}, \left\{ \begin{array}{l} x := z \\ y := c \end{array} \right.}}{\vdash \exists x \exists y. \perp, \forall z. \top} \forall \text{I}}{\quad}$$

Their interpretations yield two winning strategies $\sigma : \forall_1 1 \dashrightarrow \forall_2 \forall_3 1$ and $\tau : \forall_2 \forall_3 1 \dashrightarrow \forall_4 1$:



where we omit the annotation of negative events, forced by Σ -receptivity.

In particular, $\tau = \forall_{(\forall_2 \forall_3 1), 1}^x(\oplus_2^x \rightarrow \oplus_3^c)$. However, composing σ and τ yields $\tau \odot \sigma = \ominus_4 \rightarrow \oplus_1^c$, which cannot be of the form $\forall_{\forall_1 1, 1}^x$ as this construction would put no causal link from \ominus_4 to \oplus_1^c , since c does not involve the variable x . Hence \forall does not commute with composition, failing to be the natural bijection expected in the adjunction of a hyperdoctrine.

The intuition behind this failure is that $\forall_{A,B}^x$ only introduces causal links in a minimal way, following occurrences of a variable x . However, after composition, we may end up with elementary Σ -strategies that are not *minimal*, *i.e.* that have immediate causal links not reflecting directly a syntactic dependency. In other words, our interpretation of composition remembers more dependencies than what usual cut elimination would do: the interpretation of cut-free proofs yield minimal Σ -strategies, while in compositions interpreting cuts, causality may flow through the syntax tree of the cut formula, and create causal dependencies not reflected in the variables dependencies.

Taking the reverse point of view, we can also say that cut reduction *weakens* the causal structure of strategies.

Definition 6.11. For $\sigma, \tau : A$ two winning \mathcal{V} -strategies, σ is *weaker* than τ , written $\sigma \preceq \tau$, iff $|\sigma| \subseteq |\tau|$, $\mathcal{C}^\infty(\sigma) \subseteq \mathcal{C}^\infty(\tau)$, and for all $s \in |\sigma|$ $\lambda_\sigma(s) = \lambda_{\sigma'}(s)$.

This obviously defines a partial order which is actually equal to the partial order on elementary \mathbb{T} -strategies defined in 4.3, as inclusion of finite configuration implies inclusion over infinite configurations and Σ is a special case of inequational theory where only equal terms are related. Hence, from theorem 4.5, \preceq is a congruence with respect to \odot , \otimes and \wp . Moreover, it is a direct from its definition to see that the operation $\forall_{A,B}^x(-)$ preserves \preceq on winning strategies/ So we can actually state:

Proposition 6.3. *Let $\tau : \mathcal{B}^\perp \parallel \mathcal{C}$ a winning $(\mathcal{V} \uplus \{x\})$ -strategy and $\sigma : \mathcal{A}^\perp \parallel \mathcal{B}$ a winning \mathcal{V} -strategy, then we have:*

$$\forall_{B,C}^x(\tau) \odot \sigma \preceq \forall_{A,C}^x(\tau \odot \sigma).$$

Proof. From Lemma 6.9,

$$(\exists_{\mathcal{C}}^x) \odot (\forall_{B,C}^x(\tau) \odot \sigma) = (\tau \odot \sigma)[x/x] = (\exists_{\mathcal{C}}^x) \odot (\forall_{A,C}^x(\tau \odot \sigma))$$

hence the left and right strategies have the same causal structure and term-annotations after their \forall -prefix. These two share the same annotation (by Σ -receptivity) hence, the two strategies can only differ on which events can depend on it.

If an event depends on the \forall -prefix on the right hand side, then its labeling contains x , and so it must depend on the \forall -prefix on the left hand side as well. From that observation the inequality follows. \square

As \preceq is preserved by all operations on Σ -strategies, we deduce:

Theorem 6.3. *Let \rightsquigarrow denote the standard cut reduction in first-order MLL. Then, for two proofs π and π' of a sequent $\vdash^{\mathcal{V}} \Gamma$, we have*

$$\pi \rightsquigarrow \pi' \implies \llbracket \pi' \rrbracket^{\ell} \preceq \llbracket \pi \rrbracket^{\ell}$$

In lemma 6.3, the left and right strategies have the same terms on common events and *also* the same *events*, so only their partial order differ (see details in the proof above). In other words, the two strategies correspond to the same *expansion tree*, but differ on their acyclicity witnesses.

Unfortunately, the variant of \preceq with $|\sigma_1| = |\sigma_2|$ is not a congruence in \mathcal{V} -Games: in general, relaxing causality of σ in $\tau \odot \sigma$ may unlock new events, previously part of a causal loop. So, as it stands, we cannot generalise the above remark to any two interpretations. Yet, for MLL_1 , we conjecture that “having the same expansion tree” (*i.e.* same events and term annotations) is actually a congruence, yielding a **-autonomous hyperdoctrine*. However investigating this is left for future work.

An interpretation of LK_1

In this chapter we finally give our interpretation for the full classical sequent calculus presented in section 5.1. Unlike for MLL_1 , there is no hope of preserving its unrestricted cut reduction without collapsing to a boolean algebra [Gir91], so in the first section we only aim to map proofs to winning strategies on the appropriate game, with no care for cut elimination.

In the last section however, we investigate the computational content of LK_1 reflected in our model. In particular, we show that certain proofs with cuts lead to *infinite* strategies, closely related to the absence of strong normalisation in LK_1 . In the intermediate section we prove that this phenomenon can yet be counterbalanced by deriving a finite top-winning prefix from any infinite strategy, yielding to our compositional version of Herbrand's theorem.

7.1 Interpretation of LK_1 proofs

In this section we focus on the interpretation of proofs of the full classical sequent calculus as winning strategies in $(\mathcal{V}\text{-})\text{Games}$. From the interpretation of MLL_1 introduced in the previous chapter, what remains are the interpretation of the contraction and weakening rules.

Weakening is easy:

Lemma 7.1. *For any \mathcal{V} -games \mathcal{A} , there is a winning \mathcal{V} -strategy*

$$e_{\mathcal{A}} : \mathcal{A} \rightarrow 1.$$

For definiteness, we take $e_{\mathcal{A}}$ to be the minimal one.

Proof. The strategy $e_{\mathcal{A}}$ is simply closed under receptivity: it comprises the minimal negative events of \mathcal{A}^\perp , annotated with themselves. It is clearly winning by definition of winning on 1. \square

Contraction is much more subtle: we expect to interpret the contraction rule on a formula $\varphi \in \text{Form}(\mathcal{V})$ via some winning \mathcal{V} -strategy

$$\delta_\varphi : \llbracket \varphi \rrbracket_{\mathcal{V}} \xrightarrow{\mathcal{V}\text{-Games}} \llbracket \varphi \rrbracket_{\mathcal{V}} \otimes \llbracket \varphi \rrbracket_{\mathcal{V}}$$

Being linear, the interpretation $\llbracket - \rrbracket_{\mathcal{V}}^{\ell}$ for first order formulas presented in the previous chapter cannot validate such a requirement. A simple counter-example is the following.

Example 7.1. Consider $\varphi = \forall x.P(x)$. Then to win on

$$\exists.\neg P(\exists) \wp (\forall_1.P(\forall_1) \otimes \forall_2.P(\forall_2))$$

Player must play its \exists -move (otherwise it loses on the +-maximal configuration $\{\forall_1\}$). Furthermore, in order to win on the +-maximal configuration $\{\exists, \forall_1\}$, she must have chosen \forall_1 as witness for its \exists -move (otherwise $\neg P(\exists) \vee (\forall_1 \wedge \top)$ is not a tautology). But, the same reasoning holds on $\{\exists, \forall_2\}$ with witness \forall_2 . So Player will necessarily lose on at least one of these two +-maximal configurations.

So Player needs to be able to propagate *both* Opponent moves. For that we reinstate $?$ in the interpretation of the existential quantifier, as this was in the \exists -biased interpretation, *i.e.* $\llbracket \exists x \varphi \rrbracket_{\mathcal{V}} = ?\exists x \llbracket \varphi \rrbracket_{\mathcal{V}\wp x}$. This is not a surprise as, in classical logic, a proof (Player) may provide several witnesses for an existential quantifier. But our games model is symmetric – so that we can compose strategies – thus, for duality, we also need to allow Opponent to perform multiple attempts. We set:

Definition 7.1. To each $\varphi \in \text{Form}_{\Sigma}(\mathcal{V})$ we associate $\llbracket \varphi \rrbracket_{\mathcal{V}}$ a \mathcal{V} -game. The interpretation function $\llbracket - \rrbracket_{\mathcal{V}}$ is defined as in figure 6.1 (with \otimes considered as \wedge , 1 considered as \top and \wp considered as \vee) together with the two clauses:

$$\llbracket \exists x \varphi \rrbracket_{\mathcal{V}} = ?\exists x. \llbracket \varphi \rrbracket_{\mathcal{V}\wp \{x\}} \quad \llbracket \forall x \varphi \rrbracket_{\mathcal{V}} = !\forall x. \llbracket \varphi \rrbracket_{\mathcal{V}\wp \{x\}}$$

Having replication on \forall bélard moves means that we lose finiteness: by definition of strategies, \exists loïse must be reactive to the infinite number of copies potentially opened by \forall bélard.

This definition also means that we will need to revisit the interpretation of rules for quantifiers as the interpretation of formulas has changed. Yet we will see that it allows to interpret contraction, finding an appropriate winning \mathcal{V} -strategy $\delta_{\varphi} : \llbracket \varphi \rrbracket_{\mathcal{V}} \dashv\vdash \llbracket \varphi \rrbracket_{\mathcal{V}} \otimes \llbracket \varphi \rrbracket_{\mathcal{V}}$ for every formula $\varphi \in \text{Form}_{\Sigma}(\mathcal{V})$.

Figure 7.1 presents two simple instances of these contraction strategies (without term annotations). The first looks like the usual contraction of AJM games [AJM00]. It will be used to interpret the contraction rule on existential formulas, where it has the effect of taking the union of the different witnesses proposed by Player.

The second is less common in proof interpretations and is related to the contraction on universal formulas also in used in LK. It is however very similar to the interpretation of the parallel command of programming language

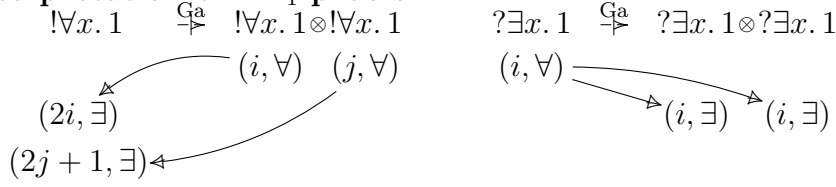


Figure 7.1: Two examples of contraction

(cf. example 4.1) : any witness proposed by \forall bélard is propagated to both branches to ensure a win.

In order to define these contraction strategies along with the tools to revisit the introduction rules for quantifiers, we first study some properties of the exponential modalities ! and ?.

7.1.1 Exponential functors

Recall the ! and ? constructions on \mathcal{V} -games from definition 5.13, they are dual enrichment with winning conditions of the countable parallel composition \parallel_ω on games (see definition 5.12). We now examine their functorial action. Being only dual on winning conditions we first introduce their common action on the structure of strategies.

Definition 7.2. Let $\sigma : A^\perp \parallel B$ be an elementary $(\Sigma \uplus \mathcal{V})$ -strategy, the *countable replication* of σ is obtained as

$$\otimes^\omega \sigma = \gamma_{A,B} * (\parallel_\omega \sigma) : (\parallel_\omega A)^\perp \parallel (\parallel_\omega B)$$

where $\gamma_{A,B} : \parallel_\omega (A^\perp \parallel B) \cong (\parallel_\omega A)^\perp \parallel (\parallel_\omega B)$ is the obvious isomorphism, and using the renaming operation of definition 4.25.

Countable parallel composition obviously preserves receptivity, courtesy, Σ -receptivity and Σ -courtesy, so $\otimes^\omega \sigma$ is an elementary $(\Sigma \uplus \mathcal{V})$ -strategy. Moreover:

Proposition 7.1. *Countable replication yields an (\preceq -order-enriched) functor on elementary $(\Sigma \uplus \mathcal{V})$ -strategies:*

$$\otimes^\omega : (\Sigma \uplus \mathcal{V})\text{-Det} \rightarrow (\Sigma \uplus \mathcal{V})\text{-Det}$$

Proof. Countable replication is a variant of the functorial action of the binary tensor \otimes in $(\Sigma \uplus \mathcal{V})\text{-Det}$, we only detail the proof for completeness.

We show the functorial equalities by studying configurations. In what follows, $x_{\sigma,i}$, $x_{\tau,i}$ respectively are configurations of σ and τ , and I stands for a finite subset of ω :

- Preservation of composition:

$$\begin{array}{ccc} \mathcal{C}(\otimes^\omega(\tau \odot \sigma)) & & \mathcal{C}((\otimes^\omega \tau) \odot (\otimes^\omega \sigma)) \\ \parallel & & \parallel \\ \{\gamma_{A,C}(\parallel_{i \in I} (x_{\tau,i} \odot x_{\sigma,i}))\} & = & \{\gamma_{A,B}(\parallel_{i \in I} x_{\tau,i}) \odot \gamma_{B,C}(\parallel_{i \in I} x_{\sigma,i})\} \end{array}$$

- Preservation of identities:

$$\begin{array}{ccc} \mathcal{C}(\otimes^\omega \mathbf{c}_A) & & \mathcal{C}(\mathbf{c}_{(\parallel_{i \in I} A)}) \\ \parallel & & \parallel \\ \{\gamma_{A,A}(\parallel_{i \in I} (x_{A,i} \sqsubseteq x'_{A,i}))\} & = & \{\parallel_{i \in I} x_{A,i} \sqsubseteq \parallel_{i \in I} x'_{A,i}\} \end{array}$$

- Preservation of the monoidal structure:

$$\begin{array}{ccc} \mathcal{C}(\otimes^\omega(\tau \otimes \sigma)) & & \mathcal{C}((\otimes^\omega \tau) \otimes (\otimes^\omega \sigma)) \\ \parallel & & \parallel \\ \{\gamma_{A \otimes B, C \otimes D}(\parallel_{i \in I} (x_{\tau,i} \otimes x_{\sigma,i}))\} & = & \{\gamma_{A,C}(\parallel_{i \in I} x_{\tau,i}) \otimes \gamma_{B,D}(\parallel_{i \in I} x_{\sigma,i})\} \end{array}$$

These equalities moreover preserve term-annotations, yielding functoriality of \otimes^ω on elementary $(\Sigma \uplus \mathcal{V})$ -strategies.

Preservation of \preceq is direct: For $\sigma \preceq \sigma' : A^\perp \parallel B$,

$$\mathcal{C}(\otimes^\omega \sigma') = \{\gamma_{A,B}(\parallel_{i \in \omega} x_{\sigma'})\} \subseteq \{\gamma_{A,B}(\parallel_{i \in \omega} x_\sigma)\} = \mathcal{C}(\otimes^\omega \sigma)$$

□

We now prove that this functor yield two functors, ! and ?, on winning \mathcal{V} -strategies,

Lemma 7.2. *Let $\sigma : \mathcal{A}^\perp \wp \mathcal{B}$ be a winning \mathcal{V} -strategy. Then,*

$$!\sigma : (!\mathcal{A})^\perp \wp !\mathcal{B} \quad ?\sigma : (? \mathcal{A})^\perp \wp ?\mathcal{B}$$

are also winning, where both are defined as $\otimes^\omega \sigma$ (definition 7.2).

Proof. This is very similar to the actions of \otimes and \wp on \mathcal{V} -strategies: let $x = (\parallel_i x_{A,i}) \parallel (\parallel_i x_{B,i}) \in \mathcal{C}^\infty(\parallel_{i \in \omega} \sigma)$ be $+$ -maximal. By definition, this implies that for all $i \in \omega$, $x_{A,i} \parallel x_{B,i} \in \mathcal{C}(\sigma)$ is $+$ -maximal in σ . Therefore, $(\mathcal{W}_A(x_{A,i})^\perp \vee \mathcal{W}_B(x_{B,i}))[\lambda_\sigma]$ is a tautology. It is then elementary to prove that both $\mathcal{W}_{(!\mathcal{A})^\perp \wp !\mathcal{B}}(x)[\lambda_{\parallel_{i \in \omega} \sigma}]$ and $\mathcal{W}_{(? \mathcal{A})^\perp \wp ?\mathcal{B}}(x)[\lambda_{\parallel_{i \in \omega} \sigma}]$ are tautologies as well. □

The functoriality of \otimes^ω is enough to induce the functoriality of both ! and ?.

7.1.2 Contraction

Rather than defining directly a contraction strategy

$$\delta_{\llbracket \varphi \rrbracket_{\mathcal{V}}} : \llbracket \varphi \rrbracket_{\mathcal{V}} \rightarrow \llbracket \varphi \rrbracket_{\mathcal{V}} \otimes \llbracket \varphi \rrbracket_{\mathcal{V}}$$

for every $\varphi \in \text{Form}_{\Sigma}(\mathcal{V})$, we start by defining several auxiliary strategies that will help in our model construction.

Dereliction. The first one is dereliction; its behaviour is the same as a copycat strategy, but it does not follow an isomorphism.

Lemma 7.3. *For any \mathcal{V} -game \mathcal{A} , there is a winning \mathcal{V} -strategy (dereliction):*

$$d_{\mathcal{A}} : !\mathcal{A} \rightarrow \mathcal{A}$$

Proof. We define this strategy as playing like copycat on \mathcal{A} between the 0-indexed copy of \mathcal{A} on the left component and the right \mathcal{A} component; other copies of \mathcal{A} on the left are simply closed under receptivity.

In particular, positive events have annotations $\lambda((1, a)^+) = (0, (0, a))^-$ and $\lambda((0, (0, a))^+) = (1, a)^-$ – negative event are annotated with their own variable, following Σ -receptivity – and configurations are of the form

$$(\|_{i \in \omega} x_i) \parallel x \in \mathcal{C}((\|^{\omega} A) \parallel A)$$

such that $x \sqsubseteq_{\mathcal{A}} x_0$, and for all $i > 0$, $\text{pol}_{\mathcal{A}}(x_i) \subseteq \{-\}$. Moreover, following the same reasoning as previously on $+$ -maximal configuration of copycat, if a configuration is $+$ -maximal in our strategy then $x_0 = x$.

From the above term-annotation, instantiating winning conditions on $(!\mathcal{A})^{\perp} \wp \mathcal{A}$ yields the formula (up to bijective renaming):

$$\left(\mathcal{W}_{\mathcal{A}}(x_0) \wedge \bigwedge_{0 < i \in \omega} \mathcal{W}_{\mathcal{A}}(x_i) \right)^{\perp} \vee \mathcal{W}_{\mathcal{A}}(x_0)$$

which is a tautology. □

Perennialization In general, it is not true that one can find a *co-dereliction* on a \mathcal{V} -game \mathcal{A} , that is, a winning strategy on $\mathcal{A} \rightarrow !\mathcal{A}$.

Example 7.2. This is very similar to example 7.1. Consider the game

$$\mathcal{A} = (\{\ominus\}, [\emptyset \mapsto \top; \{\ominus\} \mapsto \text{P}(\ominus)])$$

for some unary predicate P . If winning, a strategy $\sigma : \mathcal{A} \rightarrow !\mathcal{A}$ must play its unique positive move on the \mathcal{A} component, otherwise $x = \emptyset \parallel \{\ominus_i\}$ is

$+$ -maximal but $\mathcal{W}_{\mathcal{A}^\perp \parallel !\mathcal{A}}(x)[\lambda_\sigma] = \perp \vee P(\ominus_i)$ is not a tautology. Now if $(0, \ominus)$ is plaid then it must have an empty history as otherwise $\emptyset \parallel \{\ominus_i\}$ is $+$ -maximal, for any $(1, \ominus_i) \notin [(0, \ominus)]_\sigma^-$. So, this means that $\lambda_\sigma(0, \ominus) = \mathbf{c}$ for some constant \mathbf{c} , but, still, σ fails to win on the $+$ -maximal configuration $\{\ominus\} \parallel \{\ominus_i\}$ as the winning condition $\mathcal{W}_{\mathcal{A}^\perp \parallel !\mathcal{A}}(x)[\lambda_\sigma] = P(\mathbf{c})^\perp \vee P(\ominus_i)$ is not a tautology.

In the above, Player cannot face the exponential power given to Opponent (one can always find an Opponent move that is not in the causal history of the Player move). It turns out that for interpreted formulas, the associated \mathcal{V} -games do not face that problem; in this paragraph we show that $\llbracket - \rrbracket_{\mathcal{V}}$ yields *perennializable* \mathcal{V} -games.

Lemma 7.4. *Formulas coming from the interpretation are perennializable, i.e. for each $\varphi \in \text{Form}_\Sigma(\mathcal{V})$, there is a winning $(\Sigma \uplus \mathcal{V})$ -strategy:*

$$co_\varphi : \llbracket \varphi \rrbracket_{\mathcal{V}} \rightarrow \! \llbracket \varphi \rrbracket_{\mathcal{V}}$$

This is proved for all \mathcal{V} , by induction on formulas φ .

For units and literals, it is clear: their unique maximal configuration is the empty configuration with winning $\neg\varphi \vee (\bigwedge_{i \in \omega} \varphi)$ which is a tautology by idempotence of \wedge up to equivalence.

For other connectives, let us first exhibit some useful copycat-like winning strategies:

Lemma 7.5. *Let \mathcal{A} be a \mathcal{V} -game. Then the following isomorphisms of games preserve winning,*

$$\begin{array}{lll} !\mathcal{A} \rightarrow \! \! \mathcal{A} & !\mathcal{A} \rightarrow !\mathcal{A} \otimes !\mathcal{A} & ?!\mathcal{A} \rightarrow \! ?\mathcal{A} \\ (\langle i, j \rangle, a) \mapsto (i, (j, a)) & (2i, a) \mapsto (1, (i, a)) & (i, (j, a)) \mapsto (j, (i, a)) \\ (2i + 1, a) \mapsto (2, (i, a)) & & \\ \\ !\mathcal{A} \otimes !\mathcal{B} \rightarrow \! (\mathcal{A} \otimes \mathcal{B}) & & !\mathcal{A} \wp !\mathcal{B} \rightarrow \! (\mathcal{A} \wp \mathcal{B}) \\ (j, (i, a)) \mapsto (i, (j, a)) & & (j, (i, a)) \mapsto (i, (j, a)) \end{array}$$

Following lemma 6.1 this yields copycat winning strategies of corresponding domain and codomain.

Proof. These obviously define isomorphisms, that they preserve winning is a direct verification on the definitions of winning conditions for each constructor under study. \square

As a side remark, note that not all of these copycat strategies have an inverse that is also a winning strategy.

Back to our proof, the first isomorphism in lemma 7.5 provides a co-dereliction for $\varphi = \forall x.\psi$. For $\varphi = \varphi_1 \wedge \varphi_2$, we have

$$co_{\varphi_1 \wedge \varphi_2} = \llbracket \varphi_1 \rrbracket_{\mathcal{V}} \otimes \llbracket \varphi_2 \rrbracket_{\mathcal{V}} \mapsto !\llbracket \varphi_1 \rrbracket_{\mathcal{V}} \otimes !\llbracket \varphi_2 \rrbracket_{\mathcal{V}} \mapsto !(\llbracket \varphi_1 \rrbracket_{\mathcal{V}} \otimes \llbracket \varphi_2 \rrbracket_{\mathcal{V}})$$

using induction hypothesis, the functorial action of \otimes and the above lemma. The reasoning is the same $\varphi = \varphi_1 \wp \varphi_2$.

Finally, the only case remaining – and the most interesting one – is $\varphi = ?\exists x.\psi$. On one side, induction provides a winning strategy $? \exists x.\psi \mapsto ? \exists x.! \psi$; on the other side the right most isomorphism in lemma 7.5 yields $? ! \exists x.\psi \mapsto ! ? \exists x.\psi$. The lemma below sticks the two together by exhibiting a distribution of the existential quantifier over the exponential – modulo an infinitary explosion – yielding

$$co_{\exists x.\psi} = ? \exists x.\psi \mapsto ? \exists x.! \psi \mapsto ? ! \exists x.\psi \mapsto ! ? \exists x.\psi$$

Lemma 7.6. *Let \mathcal{A} be a $(\mathcal{V} \uplus \{x\})$ -game. Then, there is a winning $(\Sigma \uplus \mathcal{V})$ -strategy:*

$$\exists x ! \mathcal{A} \mapsto ! \exists x \mathcal{A}.$$

Proof. This strategy is defined as copycat on $! \mathcal{A}$, preceded by an infinitary trigger: once Opponent plays the existential quantifier on the left, Player simultaneously forwards it to all copies on the right, *i.e.* $\forall i > 0$, $(0, \exists) \rightarrow (1, (i, \exists))$ and $\lambda((1, (i, \exists))) = (0, \exists)$; plus (necessarily) $\lambda((1, \exists)) = (1, \exists)$.

Its $+$ -maximal configurations are thus either \emptyset , or of the form

$$(\exists.(\|_{i \in \omega} x_{A,i})) \parallel (\|_{i \in \omega} (i, \exists).y_{A,i}) \in \mathcal{C}((\exists ! A)^\perp \parallel ! \exists A)$$

with $x_{A,i} \parallel y_{A,i} \in \mathcal{C}(\mathfrak{C}_A)$ $+$ -maximal in \mathfrak{C}_A so $x_{A,i} = y_{A,i}$, for every $i \in \omega$.

The empty configuration has winning condition $\top \vee (\bigwedge_{i \in \omega} \perp)$, which is clearly tautological. Otherwise, term-annotation and winning conditions yields (a bijective renaming of) the formula

$$\left(\left(\bigwedge_{i \in \omega} \mathcal{W}_A(x_A^i) \right)^\perp \vee \left(\bigwedge_{i \in \omega} \mathcal{W}_A(x_A^i) \right) \right) [\exists/x]$$

which is also a tautology. □

Truncated comonoids Putting all of the above auxiliary strategies together, we can finally build contraction strategies for formulas' interpretation:

Corollary 7.1. *If $\varphi \in \text{Form}_\Sigma(\mathcal{V})$, then $\llbracket \varphi \rrbracket_{\mathcal{V}}$ is a “truncated comonoid” in \mathcal{V} -Games, in the sense that there are winning strategies:*

$$e_\varphi : \llbracket \varphi \rrbracket_{\mathcal{V}} \rightarrow 1 \quad \delta_\varphi : \llbracket \varphi \rrbracket_{\mathcal{V}} \rightarrow \llbracket \varphi \rrbracket_{\mathcal{V}} \otimes \llbracket \varphi \rrbracket_{\mathcal{V}}$$

Proof. The former comes from lemma 7.1, while the latter comes from the composition

$$\llbracket \varphi \rrbracket_{\mathcal{V}} \rightarrow !\llbracket \varphi \rrbracket_{\mathcal{V}} \rightarrow !\llbracket \varphi \rrbracket_{\mathcal{V}} \otimes !\llbracket \varphi \rrbracket_{\mathcal{V}} \rightarrow \llbracket \varphi \rrbracket_{\mathcal{V}} \otimes \llbracket \varphi \rrbracket_{\mathcal{V}}$$

using lemma 7.4, contraction and dereliction from lemmas 7.5 and 7.3. \square

7.1.3 Interpretation of proofs

We now conclude by providing an interpretation for all classical proofs. The interpretation is mostly informed by the interpretation of first-order MLL from the previous chapter plus the strategies for contraction and weakening defined above, however there is a small adjustment to make to the interpretation of introduction rules for quantifiers as we have changed the interpretation of the corresponding formulas by adding exponentials.

We state our final result:

Theorem 7.1. *For all \mathcal{V} , there is an interpretation $\llbracket - \rrbracket_{\mathcal{V}}$, which*

- to any $\varphi \in \text{Form}_\Sigma(\mathcal{V})$ associates a \mathcal{V} -game $\llbracket \varphi \rrbracket_{\mathcal{V}}$, and
- to any proof π of a LK sequent $\vdash^{\mathcal{V}} \varphi_1, \dots, \varphi_n$ associates a winning \mathcal{V} -strategy:

$$\llbracket \varphi \rrbracket_{\mathcal{V}} : 1 \rightarrow \llbracket \varphi_1 \rrbracket_{\mathcal{V}} \wp \dots \wp \llbracket \varphi_n \rrbracket_{\mathcal{V}}$$

Proof. We modify the interpretation of MLL by adding interpretations for contraction and weakening, and adjusting the interpretation of introduction rules for quantifiers from section 6.2.2, as shown on figure 7.2. This makes use of the (natural) isomorphism between $\Gamma^\perp \rightarrow A$ and $1 \rightarrow \Gamma \wp A$, and the (sometimes dualized) winning strategies from corollary 7.1 and lemmas 7.4 and 7.3. \square

7.2 Compositional Herbrand’s theorem

We reach now what was the first motivation of this work: giving a compositional version of Herbrand’s theorem that remains faithful to both formulas and proofs.

$$\begin{aligned}
\left[\left[\text{W} \frac{\frac{\pi}{\vdash^{\mathcal{V}} \Gamma}}{\vdash^{\mathcal{V}} \Gamma, \varphi} \right] \right] &= \Gamma^{\perp} \cong \Gamma^{\perp} \wp \perp \xrightarrow{(e_{\varphi^{\perp}})^{\perp}} \Gamma^{\perp} \wp \varphi \\
\left[\left[\text{C} \frac{\frac{\pi}{\vdash^{\mathcal{V}} \Gamma, \varphi, \varphi}}{\vdash^{\mathcal{V}} \Gamma, \varphi} \right] \right] &= \Gamma^{\perp} \xrightarrow{[\pi]} \varphi \wp \varphi \xrightarrow{\delta_{\varphi^{\perp}}^{\perp}} \varphi \\
\left[\left[\forall \text{I} \frac{\frac{\pi}{\vdash^{\mathcal{V}\wp\{x\}} \Gamma, \varphi}}{\vdash^{\mathcal{V}} \Gamma, \forall x. \varphi} \right] \right] &= \Gamma^{\perp} \xrightarrow{co_{\Gamma^{\perp}}} !\Gamma^{\perp} \xrightarrow{!(\forall([\pi]))} !\forall x. \varphi \\
\left[\left[\exists \text{I} \frac{\frac{\pi}{\vdash^{\mathcal{V}} \Gamma, \varphi[t/x]}}{\vdash^{\mathcal{V}} \Gamma, \exists x. \varphi} \right] \right] &= \Gamma^{\perp} \xrightarrow{[\pi]} \varphi[t/x] \xrightarrow{\exists_{\varphi}^t} \exists x. \varphi \xrightarrow{(d_{(\exists\varphi)^{\perp}})^{\perp}} ?\exists x. \varphi
\end{aligned}$$

Figure 7.2: Interpretation of the remaining rules of LK

In chapter 5 we intuited how wingames and winning strategies relate to the modern presentation of Herbrand's proofs that are expansion trees. We then showed that on its own, our model is well-suited to interpret first order classical proofs, with the semantic advantage of being intrinsically compositional. A direct consequence of theorem 7.1 is

Corollary 7.2. *For any closed formula φ , if $\models \varphi$ then $\llbracket \varphi \rrbracket$ has a winning strategy.*

What about its contrapositive? Is it possible to extract a proof of φ from any winning strategy $\sigma : \llbracket \varphi \rrbracket$?

As a first example, let us consider the interpretation of a proof π of $\vdash \exists x \varphi$ where φ is quantifier-free formula. This yields a winning strategy on the game $? \exists x \llbracket \varphi \rrbracket_{\{x\}}$. This game has ω positive moves, all associated to the existential quantifier. A winning strategy on this game must therefore play any subset of these moves along with annotations by some closed terms (as there are no Opponent events to provide free variables). The winning condition ensures exactly that

$$\bigvee_{(i, \exists) \in \sigma} \varphi(\lambda((i, \exists)))$$

is a tautology. This is very close to a Herbrand disjunction as presented in theorem 5.6, but is not a perfect match: there is a priori no restriction on the size of σ and it might as well be *infinite*.

In the next section (subsection 7.3.2) we will see that these strategies cannot be considered as degenerated: certain may correspond to the interpretation of some finite proofs. However, we now show that for every formulas $\varphi \in \text{Form}_\Sigma$, and every winning strategy on $\llbracket \varphi \rrbracket$, it is always possible to effectively extract a top-winning strategy on the \exists -biased interpretation $\llbracket \varphi \rrbracket^\exists$. Following expansion trees, we then prove that this defines a cut-free proof of the sequent $\vdash \varphi$. We thus finally state our compositional version of the Herbrand's theorem:

Theorem 7.2. *For any closed formula φ , the following are equivalent:*

- (1) $\vdash \varphi$,
- (2) *There exists a finite, top-winning Σ -strategy $\sigma : \llbracket \varphi \rrbracket^\exists$,*
- (3) *There exists a winning Σ -strategy $\sigma : \llbracket \varphi \rrbracket$.*

We insist that in (3), σ needs not be finite. Item (3) is our compositional statement of Herbrand's theorem: the winning strategies are those computed by our denotational model. Corollary 7.2 exactly shows that (1) implies (3), since any proof will yield by interpretation a winning Σ -strategy. We now end the discussion by showing that (3) implies (2), and (2) implies (1).

(2) implies (1): Top winning strategies yield cut-free proofs

From any finite, top-winning Σ -strategy $\sigma : \llbracket \varphi \rrbracket^\exists$, we construct a (cut-free) first-order classical proof. Contraction rules follow \exists loïse's duplications, introduction rules are inserted in an order respecting \leq_σ , and the witness of \exists I are given by λ_σ .

We show the following more general lemma:

Lemma 7.7. *For any sequent $\vdash^\nu \Gamma$, if $\sigma : \llbracket \Gamma \rrbracket_V^\exists$ is a top-winning $\Sigma \uplus \mathcal{V}$ -strategy then $\vdash^\nu \Gamma$ is provable.*

Proof. Let $|\Gamma|$ denotes the *size* of a sequent, that is, the total number of propositional connectives and quantifiers from every of its formulas. By induction on the lexicographic order $(|\sigma|, |\Gamma|)$:

- If $|\Gamma| = 0$, then Γ only consists of quantifier free formulas such that $\bigvee_{\varphi \in \Gamma} \varphi$ is a tautology. Hence, using either AX or \top I, followed by some weakening, $\vdash^\nu \Gamma$ is provable.

- If $\Gamma = \Gamma', \varphi_1 \wedge \varphi_2$ then we show that σ defines two winning strategies on $\llbracket \Gamma, \varphi_1 \rrbracket_{\mathcal{V}}^{\exists}$ and $\llbracket \Gamma, \varphi_2 \rrbracket_{\mathcal{V}}^{\exists}$ respectively. By induction these yield two proofs for $\vdash^{\mathcal{V}} \Gamma', \varphi_1$ and $\vdash^{\mathcal{V}} \Gamma', \varphi_2$ respectively, which, applying $\wedge I$, and a series of contractions, yields a proof for $\vdash^{\mathcal{V}} \Gamma$.

Let us now detail on the two strategies: they correspond to the restriction of σ to $\llbracket \Gamma, \varphi_1 \rrbracket_{\mathcal{V}}^{\exists}$ and $\llbracket \Gamma, \varphi_2 \rrbracket_{\mathcal{V}}^{\exists}$ together with its term-annotations except for the negative variables from respectively $\llbracket \varphi_2 \rrbracket^{\exists}$ and $\llbracket \varphi_1 \rrbracket^{\exists}$ that are substituted with some constant c . More formally, for $i \in \{1, 2\}$ we set $\sigma_i = \sigma_{\downarrow \llbracket \Gamma', \varphi_i \rrbracket}$, together with $\lambda_{\sigma_i} = (\lambda_{\sigma})_{|\sigma_i|} [c/\forall \in \llbracket \varphi_{3-i} \rrbracket^{\exists}]$. These obviously define elementary $\Sigma \uplus \mathcal{V}$ -strategies, top winning as – forgetting the interpretation brackets –

$$\begin{aligned} \mathcal{W}_{\Gamma}(|\sigma|)[\lambda_{\sigma}] &= (\mathcal{W}_{\Gamma'}(|\sigma_{|\Gamma'}|) \vee (\mathcal{W}_{\varphi_1}(|\sigma_{|\varphi_1}|) \wedge \mathcal{W}_{\varphi_2}(|\sigma_{|\varphi_2}|))) [\lambda_{\sigma}] \\ &\equiv ((\mathcal{W}_{\Gamma'}(|\sigma_{|\Gamma'}|) \vee \mathcal{W}_{\varphi_1}(|\sigma_{|\varphi_1}|)) \\ &\quad \wedge (\mathcal{W}_{\Gamma'}(|\sigma_{|\Gamma'}|) \vee \mathcal{W}_{\varphi_2}(|\sigma_{|\varphi_2}|))) [\lambda_{\sigma}] \\ &= \mathcal{W}_{\Gamma', \varphi_1}(|\sigma_1|)[\lambda_{\sigma}] \wedge \mathcal{W}_{\Gamma', \varphi_2}(|\sigma_2|)[\lambda_{\sigma}] \end{aligned}$$

is a tautology and thus implies that $(\mathcal{W}_{\Gamma', \varphi_1}(\sigma_1))[\lambda_{\sigma}] [c/\forall \in \llbracket \varphi_2 \rrbracket^{\exists}]$ and $(\mathcal{W}_{\Gamma', \varphi_2}(\sigma_2))[\lambda_{\sigma}] [c/\forall \in \llbracket \varphi_1 \rrbracket^{\exists}]$ also are tautological.

- Similarly, if $\Gamma = \Gamma', \forall x, \varphi$ then $\llbracket \Gamma \rrbracket_{\mathcal{V}}^{\exists} = \llbracket \Gamma' \rrbracket_{\mathcal{V}}^{\exists} \mathfrak{A} \forall. \llbracket \varphi \rrbracket_{\mathcal{V}}^{\exists}$ and, by receptivity \forall is minimal in σ . Let $\sigma' = \sigma_{\downarrow \Gamma' \mathfrak{A} \varphi}$ together with $\lambda_{\sigma'} = (\lambda_{\sigma})_{|\sigma'}| [x/\forall]$, this defines a top-winning $\mathcal{V} \uplus \{x\}$ -strategy on $\llbracket \Gamma', \varphi \rrbracket_{\mathcal{V} \uplus x}^{\exists}$. Hence by induction we get a proof for $\vdash^{\mathcal{V}, x} \Gamma', \varphi$ which extends to a proof for $\vdash^{\mathcal{V}} \Gamma$ using $\forall I$.
- Now, if $\Gamma = \Gamma', \varphi_1 \vee \varphi_2$ then σ also defines a winning strategy over $\llbracket \Gamma', \varphi_1, \varphi_2 \rrbracket_{\mathcal{V}}^{\exists}$ (the two \mathcal{V} -games are equal), so by induction and $\forall I$ this yields a proof for Γ .
- Finally, if $\Gamma = \Gamma', \exists x. \varphi$. Then either σ does not play on $\llbracket \varphi \rrbracket_{\mathcal{V}}^{\exists}$, in that case σ defines a top-winning strategy for $\llbracket \Gamma' \rrbracket_{\mathcal{V}}^{\exists}$ and by induction we have a proof for $\vdash^{\mathcal{V}} \Gamma'$ which extends to a proof for $\vdash^{\mathcal{V}} \Gamma$ using wk . Or, σ plays at least one instance \exists_i of \exists . *W.l.o.g.* we can assume that it is minimal as otherwise we fall back into one of the above cases. By minimality $\lambda_{\sigma}(\exists_i) = t \in \text{Tm}_{\Sigma}(\mathcal{V})$. Besides σ can also be viewed as playing on $\llbracket \Gamma, \exists \varphi, \exists \varphi \rrbracket_{\mathcal{V}}^{\exists}$ with every move of index i plaid on the right copy of $\exists \varphi$ and all other moves remaining in the left copy of φ . In that case, the residual strategy σ' corresponding to σ after its \exists_i move defines a top-winning \mathcal{V} -strategy on $\llbracket \Gamma', \exists \varphi, \varphi[t/x] \rrbracket_{\mathcal{V}}^{\exists}$. By induction this yields a proof for $\vdash^{\mathcal{V}} \Gamma', \exists \varphi, \varphi[t/x]$ hence for $\vdash^{\mathcal{V}} \Gamma$, using $\exists I$ followed by CTR .

□

(3) implies (2): Compactness

Finally, we show that one can extract a finite top-winning strategy from any winning strategy $\sigma : \llbracket \varphi \rrbracket$. First notice that $\llbracket \varphi \rrbracket^\exists$ embeds (subject to renaming) as a prefix of $\llbracket \varphi \rrbracket$. Keeping that renaming silent, we can restrict σ to $\llbracket \varphi \rrbracket^\exists$ by ignoring \forall bélar's replications:

Lemma 7.8. *For any winning strategy $\sigma : \llbracket \varphi \rrbracket$, setting*

$$|\sigma^\exists| = \{a \in |\sigma| \mid [a]_\sigma \subseteq \llbracket \varphi \rrbracket^\exists\}$$

and inheriting the order, polarity and labelling from σ , we obtain $\sigma^\exists : \llbracket \varphi \rrbracket^\exists$ a winning strategy.

Proof. Most conditions are straightforward. To show $\sigma^\exists : \llbracket \varphi \rrbracket^\exists$ is winning, we use that for any $+$ -maximal $x \in \mathcal{C}^\infty(\sigma^\exists)$, we have $x \in \mathcal{C}^\infty(\sigma)$ $+$ -maximal as well; indeed this follows from $\llbracket \varphi \rrbracket^\exists$ being itself $+$ -maximal in $\llbracket \varphi \rrbracket$. \square

However, the extracted σ^\exists may still not be finite; and indeed it will not always be. As mentioned earlier, the coming section 7.3.2, describes a classical proof for which our interpretation yields an infinite strategy, even after removing \forall bélar's replications.

Despite this, the compactness theorem for propositional logic entails that we can always extract a finite top-winning sub-strategy. For $\sigma : \llbracket \varphi \rrbracket^\exists$ any elementary Σ -strategy, we write $\mathcal{C}^-(\sigma)$ for the set of $--$ -maximal configurations of σ , *i.e.* they can only be extended in σ by Player moves – inheriting all structure from σ they correspond to its *sub-strategies*, as they are automatically receptive. The proof relies on:

Lemma 7.9. *Let X be a directed set of $--$ -maximal configurations. Then, $\mathcal{W}_{\llbracket \varphi \rrbracket^\exists}(\bigcup X)$ is logically equivalent to $\bigvee_{x \in X} \mathcal{W}_{\llbracket \varphi \rrbracket^\exists}(x)$.*

Proof. By induction on φ , using simple logical equivalences and that if $x_1 \subseteq x_2$ are $--$ -maximal configurations, then $\mathcal{W}_{\llbracket \varphi \rrbracket^\exists}(x_1)$ implies $\mathcal{W}_{\llbracket \varphi \rrbracket^\exists}(x_2)$. \square

We complete the proof: for $\sigma : \llbracket \varphi \rrbracket^\exists$ a winning strategy, the above lemma implies that the (potentially infinite) disjunction of finite formulas

$$\bigvee_{x \in \mathcal{C}^\forall(\sigma)} \mathcal{W}_{\llbracket \varphi \rrbracket^\exists}(x)[\lambda_\sigma]$$

is a tautology. So, by the compactness theorem there is a finite $X = \{x_1, \dots, x_n\} \subseteq \mathcal{C}^\forall(\sigma)$ such that $\bigvee_{x \in X} \mathcal{W}_{\llbracket \varphi \rrbracket^\exists}(x)[\lambda_\sigma]$ is a tautology. *W.l.o.g.* X is directed as $\mathcal{C}^\forall(\sigma)$ is closed under union so, by Lemma 7.9 again,

$\mathcal{W}_{\llbracket \varphi \rrbracket_{\exists}}(\cup X)[\lambda_{\sigma}]$ is a tautology. So, restricting σ to events $\cup X$ gives a top-winning finite sub-strategy of σ .

Although the argument above is non-constructive, the extraction of a finite sub-strategy can still be performed in an effective way: Σ -strategies and operations on them can be effectively presented, and the finite top-winning sub-strategy can be effectively obtained by Markov's principle.

7.3 Discussion on the computational content of $\llbracket - \rrbracket_{\mathcal{V}}$

In this last section, we discuss what of the computational behaviour of the classical sequent calculus is captured in our interpretation of first order classical proofs as presented in section 7.1.

Although reflecting cut reduction was not a target of this work – our main focus was on giving a compositional interpretation of Herbrand's theorem – this question remains of interest especially since, to our knowledge, our model is the first semantic model for unrestricted LK_1 .

We comment on some features of our model, leaving other questions open for future work.

7.3.1 Cut reductions

As mentioned in the introduction, classical sequent calculus with unrestricted cut reduction only has models preserving reduction boolean algebras [Gir91]. So there is no chance for our model to preserve cut reduction. Here we give a quick overview of what we know or expect of our interpretation with respect to cut reductions.

Propositional connectives Cut reductions for propositional connectives are shared by LK and MLL , so following theorem 6.1 they are faithfully reflected in our model.

Quantifiers The question for quantifiers is more tricky and we do not have a clear answer for every reduction.

The easy case remains the one of the \exists/cut rule, still valid by associativity of composition.

For \exists/\forall and \forall/cut reduction rules, one would wish to build on the result from section 6.2. Two key properties would then need to hold in our model:

naturality on co_φ , and that d_φ is the left inverse of co_φ

$$\begin{array}{ccc} \llbracket \varphi \rrbracket & \xrightarrow{co_\varphi} & !\llbracket \varphi \rrbracket \\ \downarrow \sigma & & \downarrow !\sigma \\ \llbracket \psi \rrbracket & \xrightarrow{co_\psi} & !\llbracket \psi \rrbracket \end{array} \qquad \begin{array}{ccc} \llbracket \varphi \rrbracket & \xrightarrow{co_\varphi} & !\llbracket \varphi \rrbracket \\ \Downarrow & & \downarrow d_\varphi \\ & & \llbracket \varphi \rrbracket \end{array}$$

As it stands naturality does not hold for co_φ , simply because it relies on indices that can be messed up by unrestricted strategies. Extending the present work with symmetry [CCW19] could probably solve this kind of non-matching indexes problem. However it is not clear whether that would be sufficient to prove naturality or not.

Contraction and weakening Let us now discuss the truly classical reduction rules depicted in 5.4. As for the \exists/CUT rule, the commutation rules, W/CUT and C/CUT , are preserved by associativity of composition.

Then, for the W rule, one can note that weakening strategies are natural in a weak sense.

Lemma 7.10. *For every winning \mathcal{V} -strategy $\sigma : \mathcal{A} \rightarrow \mathcal{B}$, $e_{\mathcal{A}} \preceq e_{\mathcal{B}} \odot \sigma$.*

Proof. This is immediate as every configuration in $e_{\mathcal{A}}$ is of the form $x = x_A \parallel \emptyset \in \mathcal{C}(A \parallel \mathbf{1})$ and $\text{pol}(x_A) = \{+\}$ (as $e_{\mathcal{A}}$ is receptive). But by receptivity of σ , $x_A \parallel \emptyset \in \mathcal{C}(\sigma)$ and is causally compatible with $\emptyset \parallel \emptyset \in e_{\mathcal{B}}$, so $x \in \mathcal{C}(e_{\mathcal{B}} \odot \sigma)$. Their label are obviously equal by Σ -receptivity. \square

Writing π for the left hand side of the W reduction rule and π' for its right hand-side, the lemma above yields $\llbracket \pi' \rrbracket \preceq \llbracket \pi \rrbracket$.

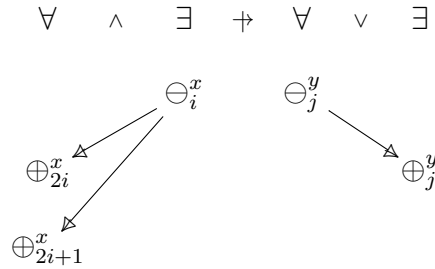
Note that there is no chance of improving that result as for example

$$\frac{\frac{\frac{\vdash \top}{\vdash \top, \perp} W \quad \frac{\frac{\frac{\vdash^x \top, \perp}{\vdash^x \top, \exists \perp} \exists I \quad \frac{\vdash \top, \forall \exists \perp}{\vdash \top, \forall \exists \perp} \forall I}{\vdash \top, \forall \exists \perp} CUT}{\vdash \top, \forall \exists \perp} W}{\vdash \top, \forall \exists \perp} W \quad \rightsquigarrow \quad \frac{\vdash \top}{\vdash \top, \forall \exists \perp} W$$

yield non-equal strategies

$$\begin{array}{ccc} \mathbf{1} & \wp & !(\forall.?(\exists. \perp)) \\ & & \forall_0 \rightarrow \exists_0^{\forall_0} \\ & & \vdots \\ & & \forall_i \rightarrow \exists_i^{\forall_i} \\ & & \vdots \end{array} \succ \begin{array}{ccc} \mathbf{1} & \wp & !(\forall.?(\exists. \perp)) \\ & & \forall_0 \\ & & \vdots \\ & & \forall_i \\ & & \vdots \end{array}$$

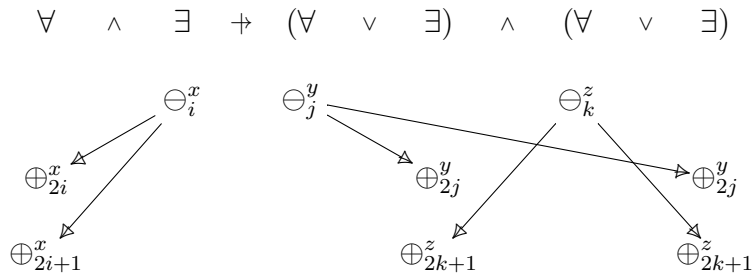
Contraction strategies however are not natural, not even in a weak sense as are weakening strategies; so the C reduction rule is not preserved by our interpretation. As an example, consider the following strategy:



This strategy is clearly definable as a proof π of *e.g.* $\vdash \exists z.Q(z) \vee \forall x.\perp, \forall y.P(y) \vee \exists z.\neg P(z)$. Now if π was cut against a proof π' of $\neg(\forall y.P(y) \vee \exists z.\neg P(z))$, ending with a contraction, then, for the ctr/cut reduction to hold, our model would have to satisfy the following naturality square:

$$\begin{array}{ccc} \forall \wedge \exists & \xrightarrow{\sigma} & \forall \vee \exists \\ \delta \downarrow & & \downarrow \delta \\ (\forall \wedge \exists) \vee (\forall \wedge \exists) & \xrightarrow{\sigma \wedge \sigma} & (\forall \vee \exists) \vee (\forall \vee \exists) \end{array}$$

But the upper right path yields the strategy:



whereas the lower left path yields:

presentation to detail, as much as reasonable, the corresponding interpretation, so that the interested reader can see it at play in a non-trivial case. We start by interpreting the structural dilemma with $\varphi = \forall x. \perp \vee \exists y. \top$, which we shorten as $\forall \vee \exists$ from now on.

Interpretation of ϖ_1 . We detail the interpretation of ϖ_1 . We start from the axioms on the left branch:

$$\left[\left[\text{Ax} \frac{}{\vdash \varphi, \varphi^\perp} \right] \right] = \begin{array}{c} (\forall \vee \exists) \quad , \quad (\exists \wedge \forall) \\ \forall_i \quad \quad \quad \forall_j \\ \quad \quad \quad \swarrow \quad \searrow \\ \quad \quad \quad \exists_j^{\forall_j} \quad \exists_i^{\forall_i} \end{array}$$

The indices i, j are the copy indices for the ! and ? arising from the interpretation of formulas, and we only display the term annotations for Eloïse's moves. The Σ -strategy above is the copycat Σ -strategy as defined in definition 2.6.

Interpreting the introduction rule for \wedge simply has the effect of tensoring two copies of copycat together, obtaining:

$$\left[\left[\text{Ax} \frac{}{\vdash \varphi, \varphi^\perp} \quad \text{Ax} \frac{}{\vdash \varphi, \varphi^\perp} \right] \right]_{\wedge \text{I}} = \left[\left[\text{Ax} \frac{}{\vdash \varphi, \varphi^\perp} \quad \text{Ax} \frac{}{\vdash \varphi, \varphi^\perp} \right] \right] =$$

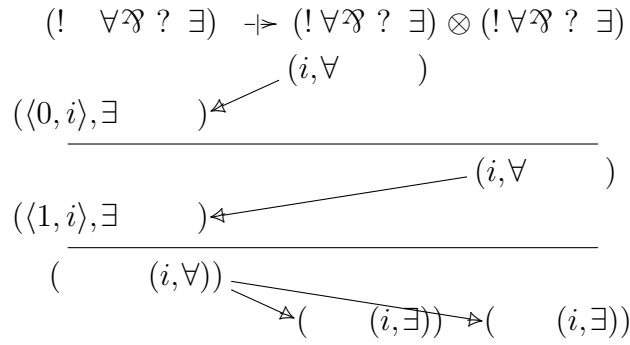
$$\begin{array}{c} (\forall \vee \exists) \wedge (\forall \vee \exists) \quad , \quad (\exists \wedge \forall) \quad , \quad (\exists \wedge \forall) \\ \forall_i \quad \quad \quad \forall_j \quad \quad \quad \forall_k \quad \quad \quad \forall_l \\ \quad \quad \quad \swarrow \quad \searrow \quad \quad \quad \swarrow \quad \searrow \\ \quad \quad \quad \exists_k^{\forall_k} \quad \exists_l^{\forall_l} \quad \exists_i^{\forall_i} \quad \exists_j^{\forall_j} \end{array}$$

i.e. again copycat, in accordance with the functoriality of \otimes .

Now, to interpret contraction, we need to compose with $\delta_{\forall \vee \exists}^\perp : (\exists \wedge \forall) \vee (\exists \wedge \forall) \rightarrow \exists \wedge \forall$, where

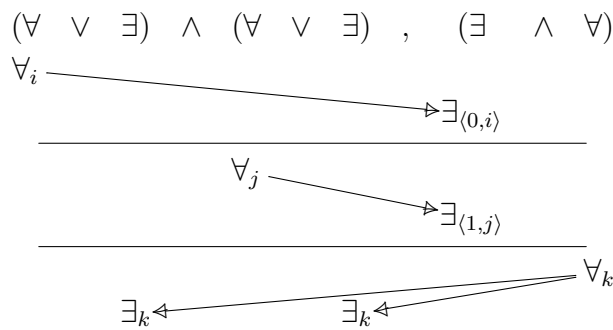
$$\delta_{\forall \vee \exists} : (!\forall \wp ?\exists) \rightarrow (!\forall \wp ?\exists) \otimes (!\forall \wp ?\exists)$$

is the contraction on φ . Note that this time, we make explicit the exponential modalities. Recall also that this strategy is derived from $co_{\forall \vee \exists} : (!\forall \wp ?\exists) \rightarrow (!\forall \wp ?\exists)$, which we display below. To display it best we deviate from the representation below by showing exactly the correspondence between copy indices and occurrences of ! and ?, and we omit the terms, which are trivial and always correspond with the unique predecessor for Eloïse's events. We display the Σ -strategy separating two sub-configurations for clarity; the full Σ -strategy is obtained by taking their union.

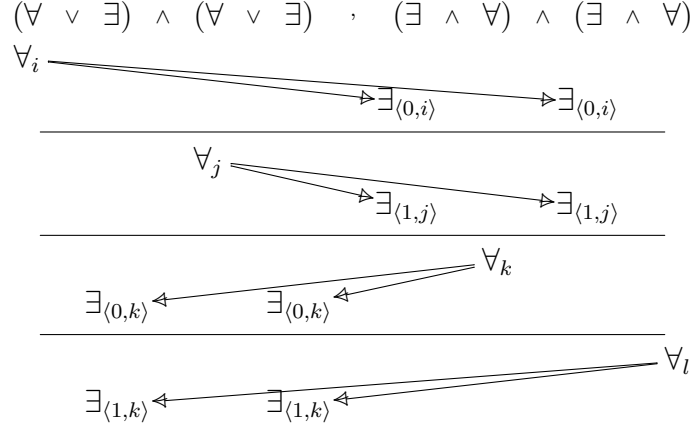


With that in place, we can finally obtain by composition (where we adopt again the simplified annotation for copy indices, since in this game ! and ? are again always attached to quantifiers – we still omit the trivial term annotations):

$$\left[\left[\frac{\text{Ax} \frac{}{\vdash \varphi, \varphi^\perp} \quad \text{Ax} \frac{}{\vdash \varphi, \varphi^\perp}}{\wedge \text{I} \frac{}{\vdash \varphi \wedge \varphi, \varphi^\perp, \varphi^\perp}} \right] \right] = \left[\frac{}{\vdash \varphi \wedge \varphi, \varphi^\perp} \right] \text{C}$$



The second branch of ϖ_1 is symmetric, so we do not make it explicit. Now, we interpret the CUT rule and the composition yields $\llbracket \varpi_1 \rrbracket$ below (again, we omit term annotations which coincide with the unique predecessor for \exists loise’s moves).

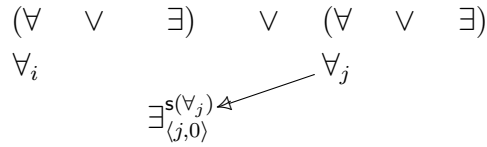


It is interesting to note that although ϖ_1 has arbitrarily large cut-free forms, the corresponding strategy only plays finitely many \exists moves for every \forall move. However, we are on the right path to finding a truly infinitary Σ -strategy.

An infinitary proof. The next step is to set (with s some unary function symbol):

$$\varpi_2 = \frac{\text{Ax} \frac{\frac{}{\vdash^x \top [s(x)/y], \perp}}{\exists I} \quad \frac{\frac{}{\vdash^x \exists y. \top, \perp}}{\forall I}}{\forall I} \quad \frac{\frac{}{\vdash \exists y. \top, \forall x. \perp}}{\text{W}}}{\forall I} \frac{\frac{}{\vdash \forall x. \perp, \exists y. \top, \forall x. \perp, \exists x. \top}}{\vdash (\forall x. \perp \vee \exists y. \top) \vee (\forall x. \perp \vee \exists x. \top)}}{\vdash (\forall x. \perp \vee \exists y. \top) \vee (\forall x. \perp \vee \exists x. \top)}$$

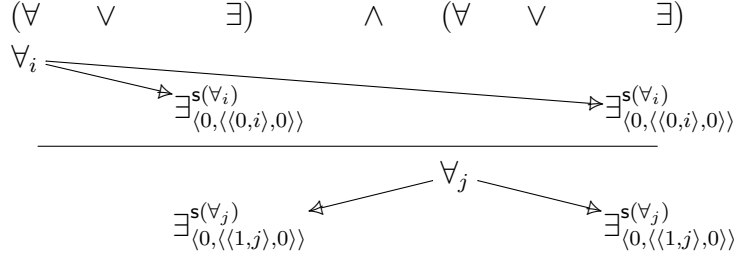
Leaving to the reader the details of the interpretation, we have by design that $\llbracket \varpi_2 \rrbracket$ is:



We now use these to compute the interpretation of:

$$\varpi_3 = \text{CUT} \frac{\frac{\varpi_1}{\vdash \varphi \wedge \varphi, \varphi^\perp \wedge \varphi^\perp} \quad \frac{\varpi_2}{\vdash (\forall \forall \exists) \vee (\forall \vee \exists)}}{\vdash \varphi \wedge \varphi}$$

The associated composition reveals $\llbracket \varpi_3 \rrbracket$ to be:



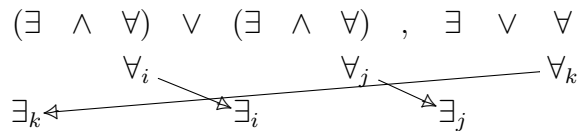
We are almost there. It suffices now to note that ϖ_3 provides a proof of

$$(\exists x. \top \implies \exists x. \top) \wedge (\exists x. \top \implies \exists x. \top)$$

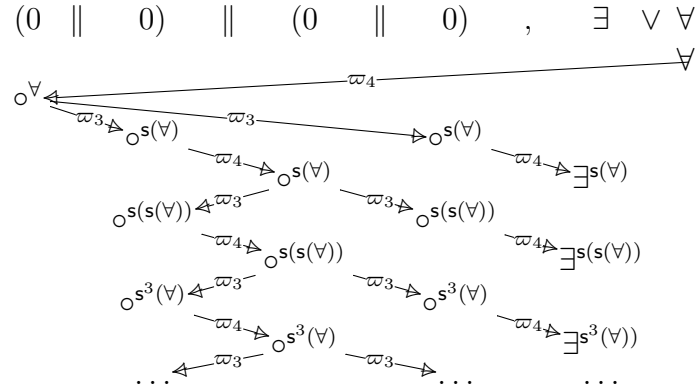
These two implications can be *composed* by cutting ϖ_3 against the proof ϖ_4 or $(\exists \implies \exists) \wedge (\exists \implies \exists) \implies (\exists \implies \exists)$ performing the composition:

$$\varpi_4 = \frac{\frac{\text{Ax} \frac{}{\vdash \forall, \exists} \quad \text{Ax} \frac{}{\vdash \forall, \exists}}{\wedge \text{I} \frac{}{\vdash \forall, \exists \wedge \forall, \exists}} \quad \text{Ax} \frac{}{\vdash \forall, \exists}}{\wedge \text{I} \frac{}{\vdash \forall, \exists \wedge \forall, \exists \wedge \forall, \exists}} \quad \text{Ex} \frac{}{\vdash \exists \wedge \forall, \exists \wedge \forall, \exists, \forall}}{\forall \text{I} \frac{}{\vdash (\exists \wedge \forall) \vee (\exists \wedge \forall), \exists \vee \forall}}$$

with interpretation:



Write ϖ_5 for the proof of $\exists x. \top \vee \forall y. \perp$ obtained by cutting ϖ_3 and ϖ_4 in the obvious way. The interpretation of ϖ_5 is the composition of $\llbracket \varpi_3 \rrbracket$ and $\llbracket \varpi_4 \rrbracket$, which triggers the feedback loop causing the infiniteness phenomenon. We display below the corresponding interaction. For the “synchronised” part of formulas, we will use 0 for components resulting from matching dual quantifiers, and \parallel for components resulting for matching dual propositional connectives. We write \circ for synchronized events (*i.e.* of neutral polarity), and omit copy indices, which get very unwieldy. For readability, we also annotate the immediate causal links with the sub-proof that they originate from, *i.e.* ϖ_3 or ϖ_4 .



Therefore, after hiding, \exists loïse responds to an initial \forall bélarde move \forall by playing simultaneously all $\exists s^n(\forall)$, for $n \geq 1$. Finally, cutting ϖ_5 against a proof of $\exists x. \top$ playing a constant symbol 0, we get a proof ϖ_6 of $\vdash \exists x. \top$ whose interpretation plays simultaneously all $\exists s^n(0)$ for $n \geq 1$.

This phenomena could certainly be avoided by adopting a *polarized* model – in the sense that formulas/games are equipped with a deterministic reduction strategies – as *e.g.* in [DJS97]. This would however means abandoning our faithfulness to the raw Herbrand content of proofs. The question of whether one could derive a non-polarized interpretation of classical first-order logic that stays finitary from the present model remains fully open. This seems to relate to the fact that syntactic notions of expansion trees with cuts [Hei10, McK13, HW13] are in general weakly, rather than strongly, normalizing.

PART III

Resource tracking concurrent games

A concurrent semantics for costs

The close correspondence between proofs and programs often reflects in the fact that models of ones are also models of the others. In part II we have seen that the concurrent games model with annotated strategies developed in part I could be used to give a semantics to first order classical proofs. In this part we present an other application of the same model, this time in the field of programming language semantics. More precisely, we provide a *quantitative* game semantics for the analysis of *costs* (or *resource usage*) of a higher-order concurrent programming language with shared state called \mathcal{R} -IPA (*Resource-tracking Idealised Parallel Algol*).

Most denotational models are *qualitative* in nature, meaning that they ignore efficiency of programs in terms of time, or other resources such as power or bandwidth. This abstraction is at the core of the methodology of denotational semantics in that it aims to capture properties that are invariant under reduction – such as termination or result of computation.

Noting that some instances of the concurrent games model with annotated strategies \mathbb{T} -CG introduced in part I are intrinsically *quantitative* – as for example \mathbb{R} -CG, presented in section 2.3.1, in which strategies carry real functions as annotations – we provide a general framework for game semantics based on concurrent games, that keeps track of *resources* as data modified throughout execution but not affecting its control flow.

Our leading example is *time*, but our construction is more general: it is parametrized by a *resource bimonoid* \mathcal{R} , an algebraic structure representing resources and their usage. This yields a sound resource-sensitive denotational model for \mathcal{R} -IPA, an affine version of the standard IPA language [GM08] with a primitive for resource consumption.

In general, our semantics is *not* adequate with respect to the parallel operational semantics we give for \mathcal{R} -IPA; our model turns out to be *more fine grained on resource usage*. Yet, our denotational semantics is not degenerate as we show that adequacy holds for an operational semantics specialized to time. Following [Ghi05], we also make use of this adequate interpretation to give insight on the semantics of programs' improvement.

Technically, our model is a refinement of the (qualitative) interpretation of *affine IPA* described in [CC16]. This interpretation is based on the concurrent games model of rigid strategies, Strat, as presented in section 4.1.1. This model has the advantage of making parallelism in computation explicit, which is essential for capturing *parallel* resource usage.

This work has been published in [ACL19].

Related work. To our knowledge, the first denotational model to cover quantitative aspects of computation such as costs was Ghica’s *slot games* [Ghi05]. They are an extension of Ghica and Murawski’s fully abstract model for a higher-order language with concurrency and shared state [GM08] that provides an interleaving semantics for the cost of execution in terms of time. Slot games exploit the intensionality of game semantics and represent time via special moves called *tokens* matching the *ticks* of a clock. They are fully abstract with respect to the notion of observation in Sands’ operational theory of *improvement* [San91].

Although close in spirit, our model differs from slot games in that our cost analysis is truly concurrent. In contrast with interleavings semantics, our analysis reflects the fact that resource consumption may combine differently in parallel and sequentially, and, in particular we can express that the simple program $\mathbf{wait}(1) \parallel \mathbf{wait}(1)$ may terminate in 1 second, rather than 2.

As a parametric model, our work is inspired from the more recent quantitative, *weighted relational model* of Laird *et al* [LMMP13]. Their model is an enrichment of the relational model of Linear Logic [Ehr12], using weights from a *resource semiring* given as parameter. This way, they capture in a single framework several notions of resources for extensions of PCF, ranging from time to probabilistic weights.

In comparison with their resource semirings $\langle \mathcal{R}, 0, 1, +, \cdot \rangle$, our *resource bimonoids* $\langle \mathcal{R}, 0, ;, \parallel, \leq \rangle$ differ however significantly: while “;” matches “ \cdot ” – the operation expressing the sequential usage of resources –, our “ \parallel ” operation – expressing the consumption of resources in parallel – is new. On the other hand, we have no counterpart for the “+”, which agglomerates distinct non-deterministically co-existing executions leading to the same values; instead our model keeps them separate. Following the collapse from the concurrent games model to the relational model [CCPW18], we expect our model to collapse to the one of Laird *et al* for an affine version of PCF. However this remains to be formally proved.

Finally, although the constructions are different, it is to note that similar parametrizations were introduced for type systems, simultaneously by, on the one hand, Ghica and Smith [GS14] and, on the other hand, Brunel, Gaboardi *et al* [BGMZ14]; the latter with a quantitative realizability denotational model.

Outline. Chapter 8 introduces the language \mathcal{R} -IPA. We sketch its interpretation in slot games and present its new parallel operational semantics together with the notion of resource bimonoids. We then recall the base

model of rigid \mathcal{R} -strategies and restrict it to negative games, constructing the denotational semantics of \mathcal{R} -IPA

In chapter 9 we focus on proving soundness for this model. We then show adequacy for an operational semantics specialized to time, noting first that the general parallel operational semantics is too coarse. Finally we give a semantic interpretation of the notion of improvement of programs in our model.

Semantics of \mathcal{R} -IPA

This chapter presents a parallel operational semantics and a concurrent game semantic for *costs/resource analysis* of a concurrent higher order programming language with shared memory.

The language corresponds to an affine version of IPA (*Idealised Parallel Algol* [GM08]) in which resource consumption is made explicit by the introduction of a special primitive **consume**. Resources are generalised as *resource bimonoids* \mathcal{R} : algebraic structures that represent resources and their consumption either sequentially or in *parallel*. In particular, we will consider resources such as time or permissions. The resulting language, called \mathcal{R} -IPA, (*Resource-tracking Idealised Parallel Algol*) is introduced in section 8.1.

In section 8.2, we focus on giving denotational semantics to this new language. We do so by enriching the concurrent games interpretation of affine IPA provided in [CC16] with \mathcal{R} -annotations. In particular, this interpretation will be carried out in a *negative restriction* of the \mathcal{R} -Strat model introduced in chapter 4, providing a cartesian symmetric monoidal closed framework for the denotation.

As in [CC16] we chose to interpret an affine language: this lets us focus on the key phenomena which are already at play, avoiding the technical hindrance caused by replication. As suggested by recent experience with concurrent games [CCPW18, CdVW19], we expect the developments presented here to extend transparently in the presence of *symmetry* [CCW15, CCW19]; this would allow us to move to the general (non-affine) setting.

8.1 Operational semantics of \mathcal{R} -IPA

This section introduces \mathcal{R} -IPA, the basic language under study in the next two chapters. It is an affine version of *Idealized Parallel Algol* (IPA) with an additional primitive for resource consumption. The operational semantics of \mathcal{R} -IPA is close to the cost model of IPA for time consumption considered by Ghica in [Ghi05], for which he provides a fully abstract interleaving-based games semantics. Our language however differs in semantics in that it account for a truly concurrent usage of time. Comparing the two semantics,

we first recall the interleaving based semantics of affine IPA with costs before presenting our operational semantics for \mathcal{R} -IPA that allows for *parallel reductions*.

8.1.1 Affine IPA

Terms and Types We start by introducing the basic language under study, *affine Idealized Parallel Algol* (IPA). It is an affine variant of the language studied in [GM08], a call-by-name concurrent higher-order language with shared state. Its *types* are given by the following grammar:

$$A, B ::= \mathbf{com} \mid \mathbf{bool} \mid \mathbf{mem}_W \mid \mathbf{mem}_R \mid A \multimap B$$

Here, \mathbf{mem}_W is the type of *writable* references and \mathbf{mem}_R is the type of *readable* references; the distinction is necessary in this affine setting as it allows to share accesses to a given state over subprocesses; this should make more sense in the next paragraph with the typing rules. In the sequel, non-functional types are called *ground types*, for which we use the notation \mathbb{X} .

We define terms directly along with their typing rules in Figure 8.1. *Contexts* are simply lists $x_1 : A_1, \dots, x_n : A_n$ of variable declarations (in which each variable occurs at most once), and the exchange rule is kept implicit. Weakening is not a rule but is admissible. We comment on a few aspects of these rules.

First, observe that the reference constructor $\mathbf{new} \ x, y \ \mathbf{in} \ M$ binds two variables x and y , one with a write permission and the other with a read permission. In this way, the permissions of a shared state can be distributed in different components of *e.g.* an application or a parallel composition, creating interferences despite the affine aspect of the language. In examples, we will often alleviate the introduction of a new state, simply writing $\mathbf{new} \ x \ \mathbf{in} \ M$ but using the subscript x_R, x_W to remind the reader of the affine nature of the two permissions.

Second, the assignment command, $M := \mathbf{tt}$, seems quite restrictive. Yet, as the language is affine, a variable can only be written to once, and, as we consider all variables initialize to \mathbf{ff} , the only useful thing to write is \mathbf{tt} .

Finally, many rules seem restrictive in that they apply only at ground type \mathbb{X} . In fact, more general rules can be defined as syntactic sugar. For instance the sequential composition ending on a function type is given by

$$M;_{A \multimap B} N = \lambda x^A. (M;_B (N x))$$

and every other construct extends via similar abstractions.

$$\begin{array}{c}
\overline{\Gamma \vdash \mathbf{skip} : \mathbf{com}} \quad \overline{\Gamma \vdash \mathbf{tt} : \mathbf{bool}} \quad \overline{\Gamma \vdash \mathbf{ff} : \mathbf{bool}} \quad \overline{\Gamma \vdash \perp : \mathbb{X}} \\
\\
\frac{(x : A) \in \Gamma}{\Gamma \vdash x : A} \quad \frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x. M : A \multimap B} \quad \frac{\Gamma \vdash M : A \multimap B \quad \Delta \vdash N : A}{\Gamma, \Delta \vdash MN : B} \\
\\
\frac{\Gamma \vdash M : \mathbf{com} \quad \Delta \vdash N : \mathbb{X}}{\Gamma, \Delta \vdash M; N : \mathbb{X}} \quad \frac{\Gamma \vdash M : \mathbf{com} \quad \Delta \vdash N : \mathbb{X}}{\Gamma, \Delta \vdash M \parallel N : \mathbb{X}} \\
\\
\frac{\Gamma \vdash M : \mathbf{bool} \quad \Delta \vdash N_1 : \mathbb{X} \quad \Delta \vdash N_2 : \mathbb{X}}{\Gamma, \Delta \vdash \mathbf{if} M N_1 N_2 : \mathbb{X}} \quad \frac{\Gamma \vdash M : \mathbf{mem}_R}{\Gamma \vdash !M : \mathbf{bool}} \\
\\
\frac{\Gamma, x : \mathbf{mem}_W, y : \mathbf{mem}_R \vdash M : \mathbb{X}}{\Gamma \vdash \mathbf{new} x, y \mathbf{in} M : \mathbb{X}} \quad \frac{\Gamma \vdash M : \mathbf{mem}_W}{\Gamma \vdash M := \mathbf{tt} : \mathbf{com}}
\end{array}$$

Figure 8.1: Typing rules for affine IPA

Examples. Despite its affine nature, the above language is non-trivial; for example it entails non-determinism.

Example 8.1. The following term is an implementation for a random coin:

$$\mathbf{coin} = (\mathbf{new} x \mathbf{in} x_W := \mathbf{tt} \parallel !x_R) : \mathbf{bool}$$

It simply reads and writes concurrently to the same memory cell.

Affine IPA can also express more complex programs such as *strictness testing* (as introduced in example 4.4):

$$\begin{aligned}
\mathbf{strict} : (\mathbf{com} \multimap \mathbf{com}) \multimap \mathbf{bool} = \\
(\lambda f^{\mathbf{com} \multimap \mathbf{com}}. \mathbf{new} x \mathbf{in} f (x_W := \mathbf{tt}); !x_R)
\end{aligned}$$

or *parallel testing*:

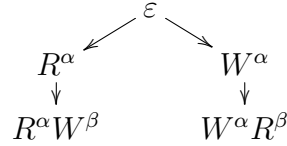
$$\begin{aligned}
&\lambda f^{\mathbf{com} \multimap \mathbf{com} \multimap \mathbf{com}}. \mathbf{new} x, y, u, v \mathbf{in} \\
&\quad f (\mathbf{if} (!x_R) (\mathbf{skip}) (u_W := \mathbf{tt}; y_W := \mathbf{tt})) \\
&\quad\quad (\mathbf{if} (!y_R) (\mathbf{skip}) (v_W := \mathbf{tt}; x_R := \mathbf{tt})); \\
&\quad\quad (!u_R) \mathbf{and} (!v_R) \\
&: (\mathbf{com} \multimap \mathbf{com} \multimap \mathbf{com}) \multimap \mathbf{bool}
\end{aligned}$$

that is, a function which on $f : \mathbf{com} \multimap \mathbf{com} \multimap \mathbf{com}$ returns true only if f returns after a concurrent call to its arguments (in the above **and** is a shorthand for the boolean “and” function $\lambda x^{\mathbf{bool}}. \lambda y^{\mathbf{bool}}. \mathbf{if} x y \mathbf{ff}$).

Memory states. The operational semantics of IPA acts on *configurations* defined as pairs $\langle M, s \rangle$ with s a *store* (yet to be defined) and $\Gamma \vdash M : A$ a term whose free variables are all of *memory types* ($\mathbf{mem}_R, \mathbf{mem}_X$).

More precisely, we fix a countable set \mathbf{L} of *memory locations*. Each location ℓ comes with two associated typed variables $\ell_R : \mathbf{mem}_R, \ell_W : \mathbf{mem}_W$ distinct from other variable names. Usually, stores are partial maps from \mathbf{L} to $\{\mathbf{tt}, \mathbf{ff}\}$. Instead, we find it more convenient for the definition of parallel reduction to introduce the notion of *state* of a memory location.

Definition 8.1. Let \mathcal{R} be an ordered set, then the *set of states over \mathcal{R}* and *accessibility relation on it*, written $(\mathbf{M}, \leq_{\mathbf{M}})$, is defined by the following *state diagram*: for every $\alpha < \beta \in \mathcal{R}$,



Ignoring \mathcal{R} and the superscripts α, β for now, a state simply corresponds to a history of memory actions (reads or writes). For each $m \in \mathbf{M}$, its set of *available actions* is

$$\text{act}(m) = \{W, R\} \setminus m$$

i.e. the letters not occurring in m – annotations being ignored; and its *value* (in $\{\mathbf{tt}, \mathbf{ff}\}$) is

$$\text{val}(m) = \begin{cases} \mathbf{tt} & \text{if } W \text{ occurs in } m, \\ \mathbf{ff} & \text{otherwise.} \end{cases}$$

Finally, a *store* is a partial map $s : \mathbf{L} \rightarrow \mathbf{M}$ with finite domain, mapping each memory location to its current state. To each store corresponds a *typing context*

$$\Omega(s) = \{\ell_X : \mathbf{mem}_X \mid \ell \in \text{dom}(s) \ \& \ X \in \text{act}(s(\ell))\}.$$

Getting back to configurations, we say that a pair $\langle M, s \rangle$ is a *valid configuration* iff M can be typed in $\Omega(s)$, meaning that every free variables in M corresponds to a memory location in the store s whose available actions allows for the variable to be used.

Interleaving based operational semantics. The one-step reductions of (call-by-name) affine IPA correspond to the basic reductions depicted on figure 8.2 together with the following contextual rule

$$\frac{\langle M, s \rangle \rightarrow \langle M', s' \rangle}{\langle \mathcal{E}[M], s \rangle \rightarrow \langle \mathcal{E}[M'], s' \rangle}$$

$$\begin{aligned}
\langle \mathbf{skip}; M, s \rangle &\rightarrow \langle M, s \rangle \\
\langle \mathbf{skip} \parallel M, s \rangle &\rightarrow \langle M, s \rangle \\
\langle M \parallel \mathbf{skip}, s \rangle &\rightarrow \langle M, s \rangle \\
\langle \mathbf{if} \mathbf{tt} N_1 N_2, s \rangle &\rightarrow \langle N_1, s \rangle \\
\langle \mathbf{if} \mathbf{ff} N_1 N_2, s \rangle &\rightarrow \langle N_2, s \rangle \\
\langle (\lambda x. M) N, s \rangle &\rightarrow \langle M[N/x], s \rangle \\
\langle !\ell_R, s \rangle &\rightarrow \langle \text{val}(s(\ell)), s[\ell \mapsto s(\ell).R] \rangle \\
\langle \ell_W := \mathbf{tt}, s \rangle &\rightarrow \langle \mathbf{skip}, s[\ell \mapsto s(\ell).W] \rangle \\
\langle \mathbf{new} x, y \mathbf{in} M, s \rangle &\rightarrow \langle M[\ell_W/x, \ell_R/y], s \uplus \{\ell \mapsto \varepsilon\} \rangle
\end{aligned}$$

Figure 8.2: Operational semantics of IPA: basic rules

where $\mathcal{E}[]$ range over the usual *call-by-name evaluation contexts* defined by:

$$\mathcal{E}[] ::= [] \mid [] N \mid []; N \mid \mathbf{if} [] N_1 N_2 \mid [] := \mathbf{tt} \mid ![] \mid ([] \parallel N) \mid (M \parallel [])$$

This set of one-step reduction rules define an *interleaving-based* operational semantics for affine IPA: a configuration $\langle M, s \rangle$ reduces to a configuration $\langle M', s' \rangle$ if there exists a reduction path $\langle M, s \rangle \rightarrow \cdots \rightarrow \langle M', s' \rangle$. The resulting rewriting system is non-deterministic, for example $\langle \mathbf{coin}, \emptyset \rangle$ reduces to two *normal forms* (*i.e.* irreducible configurations) of respective value \mathbf{tt} and \mathbf{ff} :

$$\begin{aligned}
\langle \mathbf{coin}, \emptyset \rangle &\rightarrow \langle \ell_W := \mathbf{tt} \parallel !\ell_R, [\ell \mapsto \varepsilon] \rangle \rightarrow^2 \langle !\ell_R, [\ell \mapsto W] \rangle \rightarrow \langle \mathbf{tt}, [\ell \mapsto WR] \rangle \\
\langle \mathbf{coin}, \emptyset \rangle &\rightarrow \langle \ell_W := \mathbf{tt} \parallel !\ell_R, [\ell \mapsto \varepsilon] \rangle \rightarrow^2 \langle \ell_W := \mathbf{tt} \parallel \mathbf{ff}, [\ell \mapsto R] \rangle \rightarrow^2 \langle \mathbf{ff}, [\ell \mapsto RW] \rangle
\end{aligned}$$

A game semantics Ghica and Murawski [GM08] have constructed a *fully abstract* (for may-equivalence) model for (non-affine) IPA, relying on an extension of Hyland-Ong games [HO00].

As in the above, their model takes an *interleaving* view of the execution of concurrent programs: a program is represented by the set of all its possible executions, as decided non-deterministically by the scheduler. In traditional game semantics, where execution (or *play*) are represented as sequences of moves, this is captured by removing the requirement that the two players alternate. For instance, the diagram below shows a *play* in the interpretation

of the open program $x : \mathbf{com}, y : \mathbf{bool} \vdash x \parallel y : \mathbf{bool}$:

$$\begin{array}{c}
 x : \mathbf{com}, \quad y : \mathbf{bool} \vdash \mathbf{bool} \\
 \mathbf{q}^- \\
 \mathbf{run}^+ \\
 \mathbf{q}^+ \\
 \mathbf{tt}^- \\
 \mathbf{done}^- \\
 \mathbf{tt}^+
 \end{array}$$

Contrary to concurrent games, the above diagram is read sequentially (chronologically), from top to bottom. Each line comprises one computational event (move), annotated with “−” if due to the execution environment (Opponent) and with “+” if due to the program (Player). Each move corresponds to a certain type component, under which it is placed. With the first move \mathbf{q}^- , the environment initiates the computation. Player then plays \mathbf{run}^+ , triggering the evaluation of x . In standard sequential game semantics, the control would then go back to the execution environment – Player would be stuck until Opponent plays. Here instead, due to parallelism Player can play a second move \mathbf{q}^+ immediately. At this point of execution, x and y are both running in parallel. Only when they have both returned (moves \mathbf{done}^- and \mathbf{tt}^-) is Player able to respond \mathbf{tt}^+ , terminating the computation.

The full interpretation of $x : \mathbf{com}, y : \mathbf{bool} \vdash x \parallel y : \mathbf{bool}$, is a *strategy* that comprises numerous plays like that, one for each interleaving.

8.1.2 Cost semantics and \mathcal{R} -IPA

As already mentioned, Ghica and Murawski’s model is invariant under reduction: if $\langle M, s \rangle \rightarrow \langle M', s' \rangle$, both have the same denotation. The model adequately describes the result of computation, but not its *cost* with respect to some resource consumption, as for instance with respect to time consumption, which we shall now describe.

\mathcal{R} -IPA. Consider a set \mathcal{R} of *resources*. Following [San91, Ghi05], one way to associate it with a cost semantics for IPA is to define a *cost model* assigning a cost to all basic operations. For example, every reductions on figure 8.3 would be restated as:

$$\begin{array}{l}
 \langle !\ell_R, s \rangle \rightarrow^\alpha \langle \text{val}(s(\ell)), s[\ell \mapsto s(\ell).R] \rangle \\
 \langle (\lambda x. M) N, s \rangle \rightarrow^{\alpha'} \langle M[N/x], s \rangle \\
 \dots
 \end{array}$$

for some basic costs $\alpha, \alpha', \dots \in \mathcal{R}$. To get the cost of general reduction paths, one then also needs an associative operation on \mathcal{R} , the *sequential operation* written “;”, such that for $\alpha, \beta \in \mathcal{R}$, $\alpha; \beta \in \mathcal{R}$ corresponds to the resource taken by consuming α , then β . In the case of time, one can simply choose \mathcal{R} to be the non-negative reals \mathbb{R}_+ together with addition. The total cost of a reduction path then is the sum (or the “;”-concatenation) of the costs of all basic reductions in the path.

For clarity, we choose another way to define costs on affine IPA: instead of enriching every reduction rule with costs, we separate resource consumption from the rest of the computational features and add a new construction, **consume**(α), to the language. This defines \mathcal{R} -IPA, that is, IPA plus the typing rule:

$$\frac{(\alpha \in \mathcal{R})}{\Gamma \vdash \mathbf{consume}(\alpha) : \mathbf{com}} \quad (\mathbf{consume})$$

When evaluated, **consume**(α) triggers the consumption of resource $\alpha \in \mathcal{R}$. In the case of our running example of time consumption, we will use **wait**(t), for $t \in \mathbb{R}_+$, as a synonym for **consume**(t).

To evaluate programs in \mathcal{R} -IPA, the *configurations* are now triples $\langle M, s, \alpha \rangle$ with $\alpha \in \mathcal{R}$ tracking the resources already spent. Figure 8.3 presents the basic reduction rules associated to them. One can note that the basic reduction rules of IPA presented in figure 8.2 are the same, except that their left and right hand-side now correspond to triples, for which the third resource component is left unchanged. The only rule affecting the current resources is the additional rule for **consume**(β).

Another specificity is yet to be noted: when performing memory operations the current state of resources is stored together with the action being performed, this explains the annotations in definition 8.1. These annotations do not impact the operational behaviour, but will be helpful in relating with the game semantics in Section 8.2.

Finally, in order to initialize a run, we also require $(\mathcal{R}, ;)$ to be a monoid, that is, to have a *null resource* $0 \in \mathcal{R}$ that is *neutral* for sequential composition. For time we obviously take $\mathcal{R} = (\mathbb{R}_+, 0, +)$.

Using the rules of figure 8.3 within the same call-by-name evaluation contexts as previously, there is a direct translation from IPA in a given cost model to \mathcal{R} -IPA, that is such that a program and its translation are bisimilar and reduce to the same value with the same cost. Roughly speaking, this is achieved by inserting **consume** concomitantly with the costly operations, *e.g.* :

$$!x \rightsquigarrow \mathbf{consume}(\alpha); !x$$

$$\begin{aligned}
\langle \mathbf{skip}; M, s, \alpha \rangle &\rightarrow \langle M, s, \alpha \rangle \\
\langle \mathbf{skip} \parallel M, s, \alpha \rangle &\rightarrow \langle M, s, \alpha \rangle \\
\langle M \parallel \mathbf{skip}, s, \alpha \rangle &\rightarrow \langle M, s, \alpha \rangle \\
\langle \mathbf{if} \mathbf{tt} N_1 N_2, s, \alpha \rangle &\rightarrow \langle N_1, s, \alpha \rangle \\
\langle \mathbf{if} \mathbf{ff} N_1 N_2, s, \alpha \rangle &\rightarrow \langle N_2, s, \alpha \rangle \\
\langle (\lambda x. M) N, s, \alpha \rangle &\rightarrow \langle M[N/x], s, \alpha \rangle \\
\langle !\ell_R, s, \alpha \rangle &\rightarrow \langle \text{val}(s(\ell)), s[\ell \mapsto s(\ell).R^\alpha], \alpha \rangle \\
\langle \ell_W := \mathbf{tt}, s, \alpha \rangle &\rightarrow \langle \mathbf{skip}, s[\ell \mapsto s(\ell).W^\alpha], \alpha \rangle \\
\langle \mathbf{new} x, y \mathbf{in} M, s, \alpha \rangle &\rightarrow \langle M[\ell_W/x, \ell_R/y], s \uplus \{\ell \mapsto \varepsilon\}, \alpha \rangle \\
\langle \mathbf{consume}(\beta), s, \alpha \rangle &\rightarrow \langle \mathbf{skip}, s, \alpha; \beta \rangle
\end{aligned}$$

Figure 8.3: Operational semantics: basic rules

We do not expend on that but leave [LMMP13] as a reference. Instead, we now discuss the choice of an interleaving operational semantics.

Slot Games. *Slot games* are an extension of the interleaving based model of IPA presented above to capture resource consumption [Ghi05]. These games introduce a new action $\textcircled{\$}$ called a *token* that represents an atomic resource consumption – writing \textcircled{n} for n successive occurrences of $\textcircled{\$}$.

In a model of \mathbb{N}_+ -IPA using slot games, the term ¹

$$H = (\mathbf{wait}(1); x; \mathbf{wait}(2)) \parallel (\mathbf{wait}(2); y; \mathbf{wait}(1))$$

in context $x : \mathbf{com}, y : \mathbf{bool}$, would for instance have the play depicted on figure 8.4 in its interpretation (among with many others).

Following the methodology of game semantics, the interpretation of $P = (\lambda xy. H) \mathbf{skip} \mathbf{tt}$ would then arise by composition with the interpretations of constants \mathbf{skip} and \mathbf{tt} , respectively described by their (usual) maximal plays:

$$\begin{array}{ccc}
\mathbf{com} & & \mathbf{bool} \\
\mathbf{run}^- & \text{and} & \mathbf{q}^- \\
\mathbf{done}^+ & & \mathbf{tt}^+
\end{array}$$

This would yield the strategy with only maximal play $\mathbf{q}^- \textcircled{6} \mathbf{tt}^+$, where $\textcircled{6}$ reflects the overall 6 time units (say “seconds”) that have to pass in total before we see the result (3 in each thread).

¹We use here a more liberal typing rule for ‘;’ allowing $y^{\mathbf{bool}}$; $z^{\mathbf{com}} : \mathbf{bool}$ to avoid clutter. This version of “;” can be encoded as $\mathbf{if} y(z; \mathbf{tt})(z; \mathbf{ff})$ and we will reuse this trick in other examples.

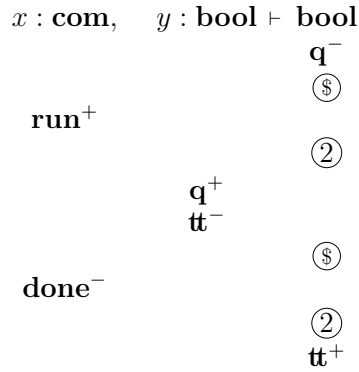


Figure 8.4: A play with tokens

This is an adequate computational analysis: the same result would be obtained by inspecting every possible derivations of P in the interleaving operational semantics described above. However, this seems wasteful; in a concurrent setting, we would expect the above term to reduce to \mathbf{tt} in only 3 seconds. The difference lies in that both slot games and the operational semantics given so far implicitly assume a *sequential* operational model, *i.e.* that both threads compete to be scheduled on a *single* processor. In the sequel, we lift that assumption and consider semantics that allow for truly parallel computations.

8.1.3 Non-interleaving operational semantics

With a truly concurrent evaluation in mind, we should be able to prove that the program P defined in the previous section may terminate in 3 seconds, rather than 6 – as nothing prevents the two threads from evaluating in parallel. Before we update the operational semantics to express that, we enrich our resource structure to allow it to express the effect of consuming resources in parallel.

Parallel resource consumption. We now introduce the full algebraic structure we require for resources.

Definition 8.2. A *resource bimonoid* is $\langle \mathcal{R}, 0, ;, \parallel, \leq \rangle$ where

- $\langle \mathcal{R}, 0, ;, \leq \rangle$ is an ordered monoid;
- $\langle \mathcal{R}, 0, \parallel, \leq \rangle$ is an ordered commutative monoid,
- 0 is bottom for \leq ,

- and \parallel is *idempotent*, *i.e.* it satisfies $\alpha \parallel \alpha = \alpha$.

A resource bimonoid is in particular a *concurrent monoid* in the sense of *e.g.* [HMSW11] – though we take \leq in the opposite direction, *i.e.* we read $\alpha \leq_{\mathcal{R}} \alpha'$ as “ α is *better/more efficient* than α' ”. Our *idempotence* assumption is rather strong as it entails that $\alpha \parallel \beta$ is the supremum of $\alpha, \beta \in \mathcal{R}$. This allows to recover a number of simple laws as, *e.g.* $\alpha \parallel \beta \leq \alpha; \beta$, stating that “parallel is better than sequential”, or the exchange rule from concurrent monoids: $(\alpha; \beta) \parallel (\alpha'; \beta') \leq (\alpha \parallel \alpha'); (\beta \parallel \beta')$. Idempotence, which would not be needed for a purely functional language, is used crucially in our interpretation of state.

Our leading example of resource is time represented by the bimonoids $\langle \mathbb{N}_+, 0, +, \max, \leq \rangle$ or $\langle \mathbb{R}_+, 0, +, \max, \leq \rangle$ – we call the latter the *time bimonoid*. Other examples are the *permission bimonoid* $\langle \mathcal{P}(P), \emptyset, \cup, \cup, \subseteq \rangle$ for some set P of *permissions*: here $(;)$ and (\parallel) are the same since, for reaching a state that requires certain permissions, it does not matter whether these permissions have been requested sequentially or in parallel; the bimonoid of *parametrized time* $\langle \mathcal{M}, 0, ;, \parallel, \leq \rangle$ with \mathcal{M} the non-decreasing functions from positive reals to positive reals, 0 the constant function, (\parallel) the pointwise maximum, and $(f; g)(x) = f(x) + g(x + f(x))$: this bimonoid tracks time consumption in a context where the time taken by **consume**(α) might grow over time.

Besides time-based bimonoids, it would be appealing to cover resources such as *power*, *bandwidth* or *heap space*. Those, however, fail idempotence of (\parallel) , and are therefore not covered. It is not clear how to extend our model to those.

Parallel operational semantics. Let us fix a resource bimonoid \mathcal{R} . To express parallel resource consumption, we move from the single-step contextual rule given in section 8.1.1 to the many-step *parallel reductions* \rightrightarrows defined in figure 8.5.

The first four rules, are simply the reflexive, transitive and contextual rules for many-steps reductions, together with the fact that a single-step reduction is a many-step reduction. Without the fifth rule (that really implements the parallel reductions), this system behaves exactly as the interleaving base semantics.

The fifth rule, for parallel composition, carries some restrictions regarding memory: M and N can only reduce concurrently if they do not access the same memory cells. This is achieved by requiring that the *partial* operation $s \uparrow s'$ – that intuitively corresponds to “merging” two memory stores s and s' whenever there are no conflicts – is defined. More formally, the partial

$$\begin{array}{c}
\frac{}{\langle M, s, \alpha \rangle \Rightarrow \langle M, s, \alpha \rangle} \text{RFL} \qquad \frac{\langle M, s, \alpha \rangle \rightarrow \langle M', s', \alpha' \rangle}{\langle M, s, \alpha \rangle \Rightarrow \langle M', s', \alpha' \rangle} \text{OTM} \\
\frac{\langle M, s, \alpha \rangle \Rightarrow \langle M', s', \alpha' \rangle}{\langle \mathcal{E}[M], s, \alpha \rangle \Rightarrow \langle \mathcal{E}[M'], s', \alpha' \rangle} \text{CTX} \\
\frac{\langle M, s, \alpha \rangle \Rightarrow \langle M', s', \alpha' \rangle \quad \langle M', s', \alpha' \rangle \Rightarrow \langle M'', s'', \alpha'' \rangle}{\langle M, s, \alpha \rangle \Rightarrow \langle M'', s'', \alpha'' \rangle} \text{TRS} \\
\frac{\langle M, s, \alpha \rangle \Rightarrow \langle M', s', \alpha' \rangle \quad \langle N, s, \alpha \rangle \Rightarrow \langle N', s'', \alpha'' \rangle}{\langle M \parallel N, s, \alpha \rangle \Rightarrow \langle M' \parallel N', s' \uparrow s'', \alpha' \parallel \alpha'' \rangle} \text{PAR}
\end{array}$$

Figure 8.5: Rules for parallel reduction

order $\leq_{\mathbf{M}}$ on memory states induces a partial order (also written $\leq_{\mathbf{M}}$) on stores, defined by $s \leq_{\mathbf{M}} s'$ iff $\text{dom}(s) \subseteq \text{dom}(s')$ and for every $\ell \in \text{dom}(s)$, $s(\ell) \leq_{\mathbf{M}} s'(\ell)$. We have

Lemma 8.1. *Two stores s_1 and s_2 have an upper bound, say s_1 and s_2 are compatible, iff for all $\ell \in \text{dom}(s_1) \cap \text{dom}(s_2)$, $s_1(\ell) \leq_{\mathbf{M}} s_2(\ell)$ or $s_2(\ell) \leq_{\mathbf{M}} s_1(\ell)$.*

In that case s_1, s_2 have a least upper bound, denoted $s_1 \uparrow s_2$.

Proof. First note that the order $\leq_{\mathbf{M}}$ on memory states is a complete partial order. So if s_1 and s_2 have an upper bound s , then, for all $\ell \in \text{dom}(s_1) \cap \text{dom}(s_2)$, $s_1(\ell) \leq_{\mathbf{M}} s(\ell)$ and $s_2(\ell) \leq_{\mathbf{M}} s(\ell)$ so $\{s_1(\ell), s_2(\ell)\}$ is directed in \mathbf{M} and either $s_1(\ell) \leq_{\mathbf{M}} s_2(\ell)$ or $s_2(\ell) \leq_{\mathbf{M}} s_1(\ell)$.

Conversely, if for all $\ell \in \text{dom}(s_1) \cap \text{dom}(s_2)$, $s_1(\ell) \leq_{\mathbf{M}} s_2(\ell)$ or $s_2(\ell) \leq_{\mathbf{M}} s_1(\ell)$ then let $s : \text{dom}(s_1) \cup \text{dom}(s_2) \rightarrow \mathbf{M}$ such that

$$s(\ell) = \max(\{s_i(\ell) \mid \ell \in \text{dom}(s_i) \wedge i \in \{1, 2\}\})$$

then s is an upper bound for s_1 and s_2 . In fact it is their least upper bound. \square

The above lemma says that if $s' \uparrow s''$ is defined in the parallel rule, then there has been no interference going to s' and s'' from their last common ancestor s . When compatible, $s' \uparrow s''$ maps s' and s'' to their least upper bound, and is undefined otherwise.

Definition 8.3. For $\vdash M : \mathbf{com}$, we say that M may converge with cost α , written $M \Downarrow_\alpha$, if $\langle M, \emptyset, 0 \rangle \Rightarrow \langle \mathbf{skip}, s, \alpha \rangle$.

For instance, instantiating the rules with the time bimonoid, we have

$$(\mathbf{wait}(1); \mathbf{wait}(2)) \parallel (\mathbf{wait}(2); \mathbf{wait}(1)) \Downarrow_3$$

\mathcal{R} -IPA with \Rightarrow enjoys a subject reduction property. More generally, writing $\alpha(m)$ for the maximal annotation on a state $m \in \mathcal{M}$ (taking 0 if $m = \varepsilon$), we say that a configuration $\langle M, s, \alpha \rangle$ is *valid* if $\langle M, s \rangle$ is valid and for every $\ell \in \text{dom}(s)$, $\alpha(s(\ell)) \leq \alpha$. We have:

Lemma 8.2. *Let $\langle M, s, \alpha \rangle$ be a valid configuration with $\Omega(s) \vdash M : A$ and $\langle M, s, \alpha \rangle \Rightarrow \langle M', s', \alpha' \rangle$, then $\langle M', s', \alpha' \rangle$ is valid with $\Omega(s') \vdash M' : A$.
Moreover, $s' = s'_1 \uplus s'_2$ such that $\text{dom}(s'_1) = \text{dom}(s)$ and $s \leq_{\mathcal{M}} s'_1$.*

Proof. By induction on the reduction tree, this is a direct check from the \rightarrow and \Rightarrow rules. \square

8.2 \mathcal{R} -strategies for \mathcal{R} -IPA

To capture the parallel resource usage of \mathcal{R} -IPA terms semantically, we build on the games model for affine IPA presented in [CC16], enriching it with \mathcal{R} -annotations, as described in chapter 4. Rather than presenting programs as collections of *sequences* of moves expressing all observable sequences of computational actions, this denotation adopts a *truly concurrent* view using rigid \mathcal{R} -strategies to represent programs, that are, collections of (annotated) *partially ordered* plays.

For each Player move, the order specifies its *causal dependencies*, *i.e.* the Opponent moves that need to have happened before. For instance, ignoring the subscripts, figure 8.6 displays a typical partially ordered play in the strategy for the term H of Section 8.1.3. Note that one partially ordered play does not fully specify a sequential execution: the one in figure 8.6 stands for *many* sequential executions, one of which is the one depicted in figure 8.4.

The behaviours expressed by partially ordered plays are deterministic *up to* choices of the scheduler – irrelevant for the eventual result. Because \mathcal{R} -IPA is non-deterministic (via concurrency and shared state), strategies are *rigid* strategies in the sens of chapter 4, that is, they are *sets* of partially order plays.

To express resources, we leverage the causal information and indicate, in each partially ordered play and for each positive move, an \mathcal{R} -expression representing its *cost* in function of the cost of its negative dependencies.

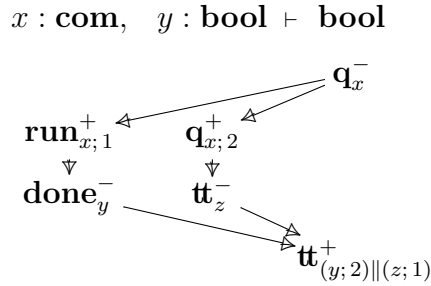
Figure 8.6: A parallel \mathcal{R} -play

Figure 8.6 displays such a \mathcal{R} -play: each Opponent move introduces a fresh variable, which can be used in the \mathcal{R} -expressions for Player moves. As we will see further on, once applied to the strategies that denote the values **skip** and **tt** (with no additional cost), this \mathcal{R} -play will answer to the initial Opponent move \mathbf{q}_x^- with $\mathbf{tt}_{x;\alpha}^+$ where $\alpha = (1; 2) \parallel (2; 1) \equiv_{\mathbb{R}_+} 3$, as prescribed by the more efficient parallel operational semantics.

We now go on to define formally our semantics.

8.2.1 Arenas and rigid \mathcal{R} -Strategies

Arenas. We first introduce *arenas*, the semantic representation of types in our model. As in [CC16], an arena will be a certain kind of *game* (see definition 1.5).

Definition 8.4. An **arena** is $(A, \leq_A, \#_A, \text{pol}_A)$, an event structure with polarities subject to:

- (i) \leq_A is forest-shaped;
- (ii) \rightarrow_A is alternating: if $a_1 \rightarrow_A a_2$, then $\text{pol}_A(a_1) \neq \text{pol}_A(a_2)$;
- (iii) it is race-free, *i.e.* if $a_1 \sim_A a_2$, then $\text{pol}_A(a_1) = \text{pol}_A(a_2)$.

Arenas present the computational actions available on a type, following a call-by-name evaluation strategy. For instance, the observable actions of a closed term on **com** are that it can be ran, and it may terminate, leading to the arena $\mathbf{com} = \mathbf{run}^- \rightarrow \mathbf{done}^+$. Likewise, a boolean can be evaluated (*i.e.* the environment is asking for a value), and it can terminate on either **tt** or **ff**. This yields the arena on the right of Figure 8.7 (when drawing arenas, immediate causality is written with a dotted line, from top to bottom).

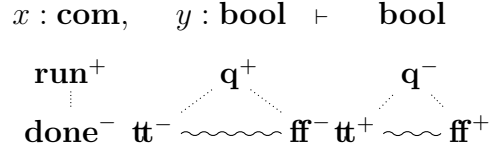


Figure 8.7: An arena for a typing judgement

Constructions on games can be restricted to arenas. Recall: the *empty arena*, written 1 , has no events; the *dual* of an arena A is A^\perp with same components, but polarities reversed; the *parallel composition* of A and B , written $A \parallel B$, has as events the tagged disjoint union $\{1\} \times A \cup \{2\} \times B$, and all other components inherited. The configurations of $A \parallel B$ are written $x_A \parallel x_B$ for $x_A \in \mathcal{C}(A)$ and $x_B \in \mathcal{C}(B)$. Figure 8.7 displays the arena $\mathbf{com}^\perp \parallel \mathbf{bool}^\perp \parallel \mathbf{bool}$.

Rigid \mathcal{R} -strategies. As hinted before, programs are going to be interpreted as rigid \mathcal{R} -strategies over arenas, that are, collections of partially ordered plays (\mathcal{R} -augmentations) with resource annotations in \mathcal{R} . Let us recall some definitions from chapter 4 specialised to \mathcal{R} -annotations, in the light of our coming interpretation.

Definition 8.5 (4.2,4.21). An *augmentation* on arena A is a finite partial order $\mathfrak{q} = (|\mathfrak{q}|, \leq_{\mathfrak{q}})$ such that $\mathcal{C}(\mathfrak{q}) \subseteq \mathcal{C}(A)$ and is *courteous*, that is, for all $a_1 \rightarrow_{\mathfrak{q}} a_2$, if $\text{pol}_A(a_1) = +$ or $\text{pol}_A(a_2) = -$, then $a_1 \rightarrow_A a_2$.

A \mathcal{R} -*augmentation* also has an *annotation* function

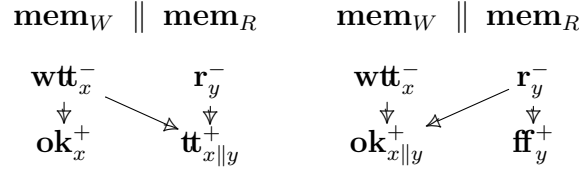
$$\lambda_{\mathfrak{q}} : (a \in |\mathfrak{q}|) \longrightarrow (\mathcal{R}^{[a]_{\mathfrak{q}}} \rightarrow \mathcal{R})$$

such that for all $a \in |\mathfrak{q}|$, $\lambda_{\mathfrak{q}}(a)$ is non-decreasing with respect to all of its variables and, if $\text{pol}_A(a) = -$, then $\lambda_{\mathfrak{q}}(a)$ is the projection on a , noted $\lambda_{\mathfrak{q}}(a)(\rho) = \rho_a$ for $\rho \in \mathcal{R}^{[a]_{\mathfrak{q}}}$.

We write $\mathcal{R}\text{-Aug}(A)$ for the set of \mathcal{R} -augmentations on A .

Recall that for $\mathfrak{q}, \mathfrak{q}' \in \mathcal{R}\text{-Aug}(A)$, \mathfrak{q} is said to be a *prefix* of \mathfrak{q}' , or that it is *rigidly embedded* in \mathfrak{q}' , written $\mathfrak{q} \hookrightarrow \mathfrak{q}'$, if $|\mathfrak{q}| \in \mathcal{C}(\mathfrak{q}')$, and for all $a, a' \in |\mathfrak{q}|$, $a \leq_{\mathfrak{q}} a'$ iff $a \leq_{\mathfrak{q}'} a'$, and $\lambda_{\mathfrak{q}}(a) = \lambda_{\mathfrak{q}'}(a)$.

The notion of \mathcal{R} -*play* is formalized by \mathcal{R} -augmentations: Figure 8.6 presents an \mathcal{R} -augmentation on the arena of Figure 8.7. The functional dependency in the annotation of positive events is represented by using the free variables introduced alongside negative events, however this is only a

Figure 8.8: Maximal \mathcal{R} -augmentations of cell

symbolic representation: the formal annotation is a function for each positive event. In the model of \mathcal{R} -IPA, we will only use the particular case where the annotations of positive events only depend on the annotations of their immediate predecessors.

Definition 8.6 (4.27). A \mathcal{R} -strategy on A is a non-empty prefix-closed set of \mathcal{R} -augmentations $\sigma \subseteq \mathcal{R}\text{-Aug}(A)$ which is *receptive*: for $q \in \sigma$ such that $|q|$ extends with $a^- \in A$ (i.e. $\text{pol}(a) = -$, $a \notin |q|$, and $|q| \cup \{a\} \in \mathcal{C}(A)$), there is $q \hookrightarrow q' \in \sigma$ such that $|q'| = |q| \cup \{a\}$.

If σ is a \mathcal{R} -strategy on arena A , we write $\sigma : A$.

As noticed in section 4.2.2, \mathcal{R} -strategies are fully described by their *maximal* augmentations, i.e. the augmentations that are the prefix of no other augmentations in the strategy.

Our interpretation of **new** will use the \mathcal{R} -strategy $\text{cell} : \mathbf{mem}_W \parallel \mathbf{mem}_R$ with arenas \mathbf{mem}_W , \mathbf{mem}_R as introduced in figure 1.4. This strategy comprises all the \mathcal{R} -augmentations rigidly included in either of the two depicted in figure 8.8. These two maximal augmentations match the race when reading and writing simultaneously: if both \mathbf{wt}^- and \mathbf{r}^- are played concurrently, the read may return \mathbf{tt}^+ or \mathbf{ff}^+ , but it can only return \mathbf{tt}^+ in the presence of \mathbf{wt}^- . Dually, if it returns \mathbf{ff}^+ , then, for sure, \mathbf{r}^- must have happen before \mathbf{wt}^- was acknowledged. At the resource level, the two possible outcomes induce different annotations: there is no additional cost on memory actions but every positive move must forward the cost of its predecessor(s).

Categorical structure As explained in sections 4.1.1 and 4.3.2, \mathcal{R} -strategies form a category: considering \mathcal{R} -strategies from A to B to be \mathcal{R} -strategies on the compound arena $A^\perp \parallel B$, they are equipped with a composition \odot and identities for it called the *copycat* \mathcal{R} -strategies (see 4.3 and 2.6). Let us recall the techniques for composition.

The composition of \mathcal{R} -strategies is based on the more primitive notion of *interactions* of \mathcal{R} -augmentations:

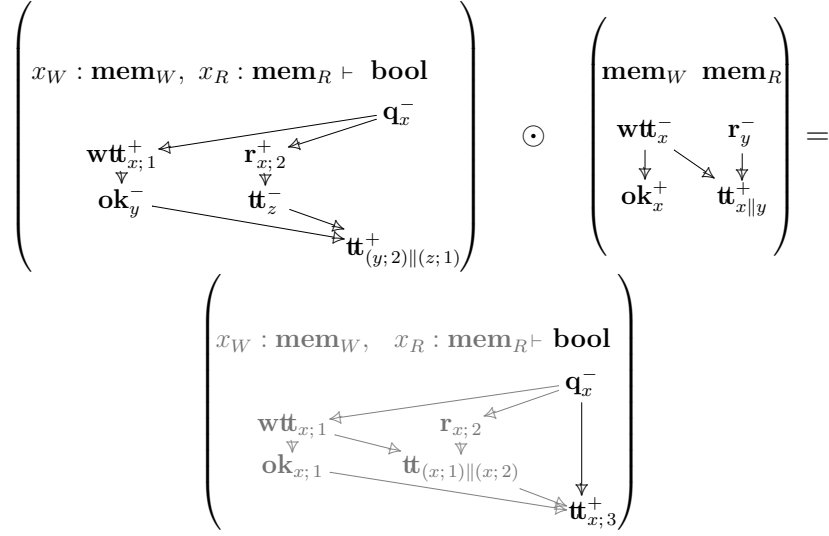


Figure 8.9: Example of interaction and composition between \mathbb{R}_+ -augmentations

Definition 8.7 (4.6,4.23). We say that $\mathfrak{q} \in \mathcal{R}\text{-Aug}(A^\perp \parallel B)$, and $\mathfrak{p} \in \mathcal{R}\text{-Aug}(B^\perp \parallel C)$ are *causally compatible* if $|\mathfrak{q}| = x_A \parallel x_B$, $|\mathfrak{p}| = x_B \parallel x_C$, and the preorder $\leq_{\mathfrak{p} \otimes \mathfrak{q}}$ on $x_A \parallel x_B \parallel x_C$ defined as $(\leq_{\mathfrak{q}} \cup \leq_{\mathfrak{p}})^+$ is a partial order.

Saying $e \in x_A \parallel x_B \parallel x_C$ is *negative* if it is negative in $A^\perp \parallel C$. We define the *interaction* $\mathfrak{p} \otimes \mathfrak{q}$ of compatible $\mathfrak{q}, \mathfrak{p}$ to be $(x_A \parallel x_B \parallel x_C, \leq_{\mathfrak{p} \otimes \mathfrak{q}}, \lambda_{\mathfrak{p} \otimes \mathfrak{q}})$ with:

$$\lambda_{\mathfrak{p} \otimes \mathfrak{q}} : (e \in x_A \parallel x_B \parallel x_C) \longrightarrow (\mathcal{R}^{[e]_{\mathfrak{p} \otimes \mathfrak{q}}} \rightarrow \mathcal{R})$$

as follows, by well-founded induction on $<_{\mathfrak{p} \otimes \mathfrak{q}}$, for $\rho \in \mathcal{R}^{[e]_{\mathfrak{p} \otimes \mathfrak{q}}}$:

$$\lambda_{\mathfrak{p} \otimes \mathfrak{q}}(e)(\rho) = \begin{cases} \lambda_{\mathfrak{p}}(e) \left(\langle \lambda_{\mathfrak{p} \otimes \mathfrak{q}}(e')(\rho) \mid e' \in [e]_{\mathfrak{p}}^- \rangle \right) & \text{if } \text{pol}_{B^\perp \parallel C}(e) = +, \\ \lambda_{\mathfrak{q}}(e) \left(\langle \lambda_{\mathfrak{p} \otimes \mathfrak{q}}(e')(\rho) \mid e' \in [e]_{\mathfrak{q}}^- \rangle \right) & \text{if } \text{pol}_{A^\perp \parallel B}(e) = +, \\ \rho_e & \text{otherwise (} e \text{ negative).} \end{cases}$$

If $\sigma : A^\perp \parallel B$ and $\tau : B^\perp \parallel C$, we write $\tau \otimes \sigma$ for the set comprising all $\mathfrak{p} \otimes \mathfrak{q}$ such that $\mathfrak{p} \in \tau$ and $\mathfrak{q} \in \sigma$ are causally compatible. For $\mathfrak{q} \in \sigma$ and $\mathfrak{p} \in \tau$ causally compatible with $|\mathfrak{p} \otimes \mathfrak{q}| = x_A \parallel x_B \parallel x_C$, their *composition* is $\mathfrak{p} \odot \mathfrak{q} = (x_A \parallel x_C, \leq_{\mathfrak{p} \odot \mathfrak{q}}, \lambda_{\mathfrak{p} \odot \mathfrak{q}})$ where $\leq_{\mathfrak{p} \odot \mathfrak{q}}$ and $\lambda_{\mathfrak{p} \odot \mathfrak{q}}$ are the restrictions of $\leq_{\mathfrak{p} \otimes \mathfrak{q}}$ and $\lambda_{\mathfrak{p} \otimes \mathfrak{q}}$. Similarly to the interaction, the *composition* of two \mathcal{R} -strategies $\sigma : A^\perp \parallel B$ and $\tau : B^\perp \parallel C$ is the set comprising all $\mathfrak{p} \odot \mathfrak{q}$ for $\mathfrak{q} \in \sigma$ and $\mathfrak{p} \in \tau$ causally compatible.

In Figure 8.9, we illustrate the construction of composition between \mathbb{R}_+ -augmentations – with also in gray the underlying interaction. The reader

may check that the variant of the left \mathbb{R}_+ -augmentation with \mathbf{tt} replaced with \mathbf{ff} is causally compatible with the other augmentation in Figure 8.8, with composition $\mathbf{q}_x^- \rightarrow \mathbf{ff}_{x,4}^+$.

Finally there is a tensor operation: on arenas, $A \otimes B$ is simply a synonym for $A \parallel B$; on \mathcal{R} -strategies it is defined componentwise from the *tensor product* on \mathcal{R} -augmentations: $\mathfrak{q}_1 \otimes \mathfrak{q}_2 \in \mathcal{R}\text{-Aug}((A_1 \otimes A_2)^\perp \parallel (B_1 \otimes B_2))$ for $\mathfrak{q}_1 \in \mathcal{R}\text{-Aug}(A_1^\perp \parallel B_1)$ and $\mathfrak{q}_2 \in \mathcal{R}\text{-Aug}(A_2^\perp \parallel B_2)$.

Following theorem 4.3, we have:

Proposition 8.1. *There is a compact closed category $\mathcal{R}\text{-Strat}$ having arenas as objects, and as morphisms, \mathcal{R} -strategies between them.*

8.2.2 Negative interpretation of \mathcal{R} -IPA

Negative Arenas and \mathcal{R} -Strategies. As a compact closed category, $\mathcal{R}\text{-Strat}$ is a model of the linear λ -calculus. However, we will (as usual for call-by-name) instead interpret \mathcal{R} -IPA in a sub-category of *negative* arenas and strategies, in which the empty arena 1 is terminal, providing the interpretation of weakening. This sub-category also has products, providing an interpretation for **if**. We will stay brief here, as this proceeds exactly as in [CC16, Cas17].

We say that a partial order with polarities is *negative* if all its minimal events are. This applies in particular to arenas, and \mathcal{R} -augmentations. By extension, a \mathcal{R} -strategy is *negative* if all its \mathcal{R} -augmentations are. A negative \mathcal{R} -augmentation $\mathfrak{q} \in \mathcal{R}\text{-Aug}(A)$ is *well-threaded* if for all $a \in |\mathfrak{q}|$, $[a]_{\mathfrak{q}}$ has exactly a one minimal event. Similarly, a \mathcal{R} -strategy is *well-threaded* iff all its \mathcal{R} -augmentations are. The reader may check that every example presented so far in this chapter depicts well-threaded \mathcal{R} -strategies. Moreover, as arenas are forest-shaped, all of their events have a unique minimal event in their causal history, this implies that the copycat strategies on negative arenas are also well-threaded. We have:

Proposition 8.2. *Negative arenas and well-threaded \mathcal{R} -strategies form a cartesian symmetric monoidal closed category $\mathcal{R}\text{-Strat}_-$, with 1 terminal.*

We also write $\sigma : A \rightarrow B$ for morphisms in $\mathcal{R}\text{-Strat}_-$.

Recall that a monoidal category (\mathcal{C}, \otimes) is *closed* if for every object $A \in \mathcal{C}$, the functor $_ \otimes A : \mathcal{C} \rightarrow \mathcal{C}$ has a right adjoint, called its *internal-hom*. Compact closed categories (see page 37) are particular cases of monoidal closed categories where the internal-hom is defined by $A^* \otimes _$ for A^* the dual object of A .

Negative arenas are closed under parallel composition (hence tensor). However, the dual operation $(-)^{\perp}$ is not well-defined on negative arenas, that is why the compact closure of $\mathcal{R}\text{-Strat}$ does not transport to $\mathcal{R}\text{-Strat}_{\perp}$. Yet, we can replace the internal-hom of a negative arena A , $A^{\perp} \parallel _$, in $\mathcal{R}\text{-Strat}$ by a negative version of it, written $A \multimap _$. Here we describe only a restricted case of the general construction in [CC16], which is however sufficient for the types of \mathcal{R} -IPA.

Definition 8.8. If A, B are negative arenas and B is *well-opened*, i.e. it has exactly one minimal event b , we form $A \multimap B$ as having all components as in $A^{\perp} \parallel B$, with additional dependencies $\{(2, b), (1, a) \mid a \in A\}$. For the empty arena, we set $A \multimap \mathbf{1} = \mathbf{1}$.

One can note that strategies over $A \multimap B$ are necessarily well-threaded as the arena as at most one minimal event. Moreover, the set of (well-threaded) strategies over $A \multimap B$ is equal to the set of well-threaded strategies over $A^{\perp} \parallel B$ as this arena has at most one negative event that is minimal, hence, that must be the minimal event of every move in a well-threaded strategy. More generally, for B well-opened, this allows to type any well-threaded strategy $\sigma : C \otimes A \multimap B$ as a well-threaded strategy from C to $A \multimap B$, then written $\Lambda(\sigma) : C \multimap A \multimap B$. Using the compact closed structure of $\mathcal{R}\text{-Strat}$, it is then easy to build a copycat \mathcal{R} -strategy $\text{ev}_{A,B} : (A \multimap B) \otimes A \multimap B$, the evaluation strategy of the monoidal closure.

Restricted to negative arenas, one can define the *cartesian product* of A and B , written $A \& B$, as having the same components as $A \parallel B$, except for the conflict relation which is extended with

$$(1, a) \# (2, b) \quad \text{for all } a \in A, b \in B$$

We write $\pi_i : A_1 \& A_2 \multimap A_i$ for the copycat projections, and $\langle \sigma, \tau \rangle : A \multimap B \& C$ for the pairing of $\sigma : A \multimap B$, and $\tau : A \multimap C$ (simply defined as $\sigma \cup \tau$ with the appropriate renaming of events from B and C).

Finally, for any A , we write $!_A : A \multimap \mathbf{1}$ for the unique (empty) such well-threaded \mathcal{R} -strategy.

Interpretation Compared to [CC16], there is not much change between the interpretation of affine IPA and \mathcal{R} -IPA. Types are interpreted equally with: base type **com**, **bool** respectively interpreted as in the left and right-hand side of figure 8.7 (with reverse polarities for **com**); **mem_W** and **mem_R** as in figure 1.4; and $\llbracket A \multimap B \rrbracket = \llbracket A \rrbracket \multimap \llbracket B \rrbracket$, as expected. Then, contexts $\Gamma = x_1 : A_1, \dots, x_n : A_n$ are interpreted as tensor products of their components: $\llbracket \Gamma \rrbracket = \otimes_{1 \leq i \leq n} \llbracket A_i \rrbracket$.

$$\begin{aligned}
\llbracket \lambda x. M : A \multimap B \rrbracket &= \Lambda(\llbracket M \rrbracket) \\
\llbracket M^{A \multimap B} N : B \rrbracket &= \text{ev}_{A,B} \odot (\llbracket M \rrbracket \otimes \llbracket N \rrbracket) \\
\llbracket M ; N : \mathbb{X} \rrbracket &= \text{seq}_{\mathbb{X}} \odot (\llbracket M \rrbracket \otimes \llbracket N \rrbracket) \\
\llbracket M \parallel N : \mathbb{X} \rrbracket &= \text{par}_{\mathbb{X}} \odot (\llbracket M \rrbracket \otimes \llbracket N \rrbracket) \\
\llbracket \text{if } M N_1 N_2 : \mathbb{X} \rrbracket &= \text{if}_{\mathbb{X}} \odot (\llbracket M \rrbracket \otimes \langle \llbracket N_1 \rrbracket, \llbracket N_2 \rrbracket \rangle) \\
\llbracket !M : \mathbf{bool} \rrbracket &= \text{deref} \odot \llbracket M \rrbracket \\
\llbracket M := \mathbf{tt} : \mathbf{com} \rrbracket &= \text{assign} \odot \llbracket M \rrbracket \\
\llbracket \text{new } x, y \text{ in } M : \mathbb{X} \rrbracket &= \llbracket M \rrbracket \odot (\llbracket \Gamma \rrbracket \otimes \text{cell})
\end{aligned}$$

Figure 8.10: Interpretation of \mathcal{R} -IPA

For terms, the interpretation of $\Gamma \vdash M : A$ is performed inductively by mapping to rigid \mathcal{R} -strategies $\llbracket M \rrbracket : \llbracket \Gamma \rrbracket \rightarrow \llbracket A \rrbracket$ (we will often omit the interpretation brackets around types).

For constant, the basic strategies are:

- $\llbracket \perp \rrbracket : \mathbf{com}$, the diverging \mathcal{R} -strategy with no player move;
- $\llbracket \text{consume}(\alpha) \rrbracket : \mathbf{com}$, the strategy of maximal \mathcal{R} -augmentation $\mathbf{run}_x^- \rightarrow \mathbf{done}_{x;\alpha}^+$, that is, is the strategy that returns with an extra cost of α ;
- $\llbracket \text{skip} \rrbracket : \mathbf{com}$, simply equal to $\llbracket \text{consume}(0) \rrbracket$, the neutral consumption for sequential and parallel composition;
- $\llbracket \mathbf{tt} \rrbracket$ and $\llbracket \mathbf{ff} \rrbracket : \mathbf{bool}$, the constant \mathcal{R} -strategies of maximal augmentation $\mathbf{q}_x^- \rightarrow \mathbf{tt}_x^+$ and $\mathbf{q}_x^- \rightarrow \mathbf{ff}_x^+$ respectively.

These strategies are pre-composed by the weakening strategy $!_{\Gamma}$ to associate them with the right context.

The rest of the interpretation is given in figure 8.10, it follows the standard interpretation of the affine λ -calculus together with using the \mathcal{R} -strategy cell introduced in Figure 8.8 and the additional \mathcal{R} -strategies with maximal \mathcal{R} -augmentations in Figure 8.11.

8.2.3 Pre-orders on \mathcal{R} -strategies

Not needed in the mere interpretation of \mathcal{R} -IPA, we conclude this section by introducing two pre-orders on \mathcal{R} -strategies that generalise the order on \mathcal{R} and will be needed in the sequel.

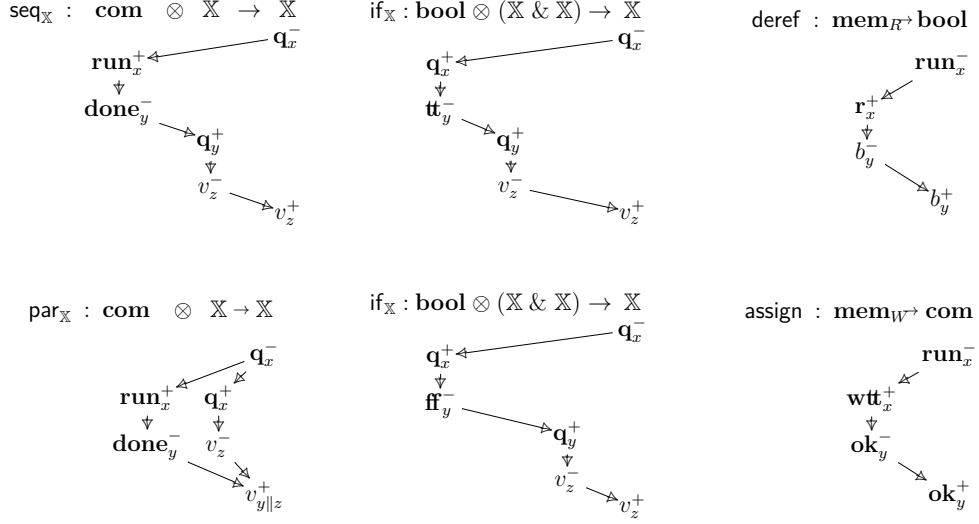


Figure 8.11: Maximal \mathcal{R} -augmentations of \mathcal{R} -strategies used in the interpretation

Partial orders on \mathcal{R} -augmentations As usual we first start by examining the case on \mathcal{R} -augmentations. \mathcal{R} -augmentations already enjoy a partial order: let $q, q' \in \mathcal{R}\text{-Aug}(A)$, we write $q \preceq q'$ iff $|q| = |q'|$ and $\leq_q \subseteq \leq_{q'}$, that is, q and q' share the same events but q is less causally constrained. As shown in section 4.1.1 and lemma 4.3 this partial-order is preserved by \odot and \otimes on \mathcal{R} -augmentations (note that in this section we actually use the sign \preceq in the reversed way).

Then there are two possible ways to lift the order of \mathcal{R} to augmentations:

- either by building on top of \preceq , setting $q \preceq_{\mathcal{R}} q'$ if $q \preceq q'$ and for every $a \in q$, $\lambda_q(a) \leq_{\mathcal{R}} \lambda_{q'}(a)$;
- or by keeping the lift of \mathcal{R} on annotations separated from the rest of the structure, only comparing two \mathcal{R} -augmentations if they share the same underlying plain augmentation, that is, setting $q \leq_{\mathcal{R}} q'$ if $q = q'$ as partial order and for every $a \in q$, $\lambda_q(a) \leq_{\mathcal{R}} \lambda_{q'}(a)$.

Obviously $\leq_{\mathcal{R}} \subseteq \preceq_{\mathcal{R}}$ and they both define a partial-order. Moreover, by lemmas 4.18 and 4.19 they are both preserved by composition and tensor on \mathcal{R} -augmentations.

Pre-orders on \mathcal{R} -strategies Lifting the partial order $\preceq_{\mathcal{R}}$, $\leq_{\mathcal{R}}$ from \mathcal{R} -augmentations on A to \mathcal{R} -strategies on A componentwise can be done in

many ways, depending on the choice made to relate the \mathcal{R} -augmentations of one strategy to the \mathcal{R} -augmentations of the strategy it is compared to.

For $\preceq_{\mathcal{R}}$ we chose a lax relation: given two \mathcal{R} -strategies $\sigma, \tau : A$, we write $\sigma \preceq_{\mathcal{R}} \tau$ if for every $\mathfrak{q} \in \tau$ there exists $\mathfrak{q}' \in \sigma$ such that $\mathfrak{q}' \preceq_{\mathcal{R}} \mathfrak{q}$. This will provide a way to formalise the fact that a strategy (σ) “improves” an other strategy (τ) as it allows for more interactions with lesser resource consumption. This only defines a pre-order on \mathcal{R} -strategies as not every augmentation in σ needs to be an “improvement” of an augmentation in τ . Conversely, an augmentation in σ can improve many augmentations in τ .

For example

$$\begin{array}{ccc} \left[\begin{array}{l} \text{new } x \text{ in} \\ \text{consume}(\alpha); \parallel \text{consume}(\beta); \\ x_W := \mathbf{tt} \end{array} \right] & & \left[\begin{array}{l} \text{new } x \text{ in} \\ \text{consume}(\alpha); !x_R; \\ \text{consume}(\beta); x_W := \mathbf{tt} \end{array} \right] \\ \{ \mathfrak{q}_x^-, \mathfrak{q}_x^- \rightarrow \mathbf{tt}_{x;(\alpha\parallel\beta)}^+, \mathfrak{q}_x^- \rightarrow \mathbf{ff}_{x;(\alpha\parallel\beta)}^+ \} & \preceq_{\mathcal{R}} & \{ \mathfrak{q}_x^-, \mathfrak{q}_x^- \rightarrow \mathbf{ff}_{x;\alpha;\beta}^+ \} \end{array}$$

For $\leq_{\mathcal{R}}$, we also choose a lax relation but we keep the focus on annotations so we set: $\sigma \leq_{\mathcal{R}} \tau$ if $\sigma = \tau$ as rigid strategies and for every $\mathfrak{q}' \in \tau$ there exists $\mathfrak{q} \in \sigma$ such that $\mathfrak{q} \leq_{\mathcal{R}} \mathfrak{q}'$. This again defines a pre-order on \mathcal{R} -strategies, but a more constrained one: σ interacts the same way as τ does, only more efficiently. For example

$$\begin{array}{ccc} \left[\text{consume}(\alpha) \parallel \text{consume}(\beta) \right] & & \left[\text{consume}(\alpha); \text{consume}(\beta); \right] \\ \{ \text{run}_x^-, \text{run}_x^- \rightarrow \text{done}_{x;(\alpha\parallel\beta)}^+ \} & \leq_{\mathcal{R}} & \{ \text{run}_x^-, \text{run}_x^- \rightarrow \text{done}_{x;\alpha;\beta}^+ \} \end{array}$$

This also characterises an “improvement” (and we have $\leq_{\mathcal{R}} \subseteq \preceq_{\mathcal{R}}$) but the first use we will make of $\leq_{\mathcal{R}}$ is actually in the proofs of soundness of \mathcal{R} -IPA, showing that if $\langle M, s, \alpha \rangle \Rightarrow \langle M', s', \alpha' \rangle$ then $\llbracket \langle M, s, \alpha \rangle \rrbracket \leq_{\mathcal{R}} \llbracket \langle M', s', \alpha' \rangle \rrbracket$ for an interpretation $\llbracket - \rrbracket$ on configurations yet to be given.

Before moving to this definition, we note that, as a consequence of the preservation of $\preceq_{\mathcal{R}}$ and $\leq_{\mathcal{R}}$ by \odot and \otimes on \mathcal{R} -augmentations, the pre-orders $\preceq_{\mathcal{R}}$ and $\leq_{\mathcal{R}}$ on \mathcal{R} -strategies are also preserved by the componentwise composition and tensor product on \mathcal{R} -strategies.

Soundness and Adequacy

In this chapter we show that the interpretation of \mathcal{R} -IPA in the concurrent games model of rigid \mathcal{R} -strategies as presented in chapter 8 is sound. Adequacy however does not hold in general, the operational semantics induces some artificial synchronisations that increase resource-consumption in comparison to what is expected from the denotational semantics. In section 9.2, we give credit to the game semantics, by showing that in the case of time, the operational semantics can be refined to get adequacy back. Inspired by [Ghi05], we finally present how the resulting sound and adequate interpretation of \mathbb{R}^+ -IPA (*i.e.* \mathcal{R} -IPA with the specialised operational semantic for time) provides insight in the understanding of *improvements* of programs in a truly concurrent setting.

9.1 Soundness for \mathcal{R} -IPA

In this section, we set to prove that the game semantics of \mathcal{R} -IPA described in section 8.2.2 is sound with respect to its operational semantics given in section 8.1.3.

We first introduce a useful notation. For any type A , $\llbracket A \rrbracket$ has a unique minimal event; we write $\langle A \rangle$ for the arena without this minimal event. Likewise, if $\Gamma \vdash M : A$, then by construction, $\llbracket M \rrbracket : \llbracket \Gamma \rrbracket^\perp \parallel \llbracket A \rrbracket$ is a well-threaded \mathcal{R} -strategy whose augmentations all share the same minimal event $\mathbf{q}_{A,x}^-$ where \mathbf{q}_A^- is minimal in A and x is the variable name attached to \mathbf{q}_A^- for convenience in writing function annotations in augmentations. For $\alpha \in \mathcal{R}$, we define $\langle M \rangle_\alpha : \llbracket \Gamma \rrbracket^\perp \parallel \langle A \rangle$ the \mathcal{R} -strategy having the same augmentations as $\llbracket M \rrbracket$ but without $\mathbf{q}_{A,x}^-$, whose corresponding variable x is then replaced by α . One may think of $\langle M \rangle_\alpha$ as “ M started with consumed resource α ”.

Naively, one may expect soundness to state that for all $\vdash M : \mathbf{com}$, if $M \Downarrow_\alpha$, then $\langle M \rangle_0 = \mathbf{done}_\alpha^+$. However, whereas the resource annotations in the denotations are always as good as permitted by the causal constraints, derivations in the operational semantics may be sub-optimal. For instance, we may derive $M \Downarrow_\alpha$ not using the parallel rule at all. So our statement is:

Theorem 9.1. *If $\vdash M : \mathbf{com}$ with $M \Downarrow_\alpha$, then, there is $\beta \leq_{\mathcal{R}} \alpha$ s.t. $\mathbf{done}_\beta^+ \in \langle M \rangle_0$.*

Our proof methodology is standard: we replay operational derivations as augmentations in the denotational semantics. Stating the invariant successfully proved by induction on operational derivations requires some technology.

9.1.1 Interpretation of memory states

Configurations of \mathcal{R} -IPA are of the form $\langle M, s, \alpha \rangle$ with $\Omega(s) \vdash M : A$. In our game semantics the data of M and α is translated as $\langle M \rangle_\alpha : \llbracket \Omega(s) \rrbracket \rightarrow \langle A \rangle$; we also need to make sense of the store s .

cell strategies. Following the interpretation of **newref**, where a single (fresh) memory location is interpreted by the strategy **cell**, a memory store s is associated to a memory strategy $\mathbf{cell}_s : \llbracket \Omega(s) \rrbracket$. More precisely,

$$\mathbf{cell}_s = \otimes_{\ell \in \text{dom}(s)} \mathbf{cell}_{s(\ell)}$$

for $\mathbf{cell}_{s(\ell)}$ the \mathcal{R} -strategies given by:

- $\mathbf{cell}_\varepsilon = \mathbf{cell}$;
- \mathbf{cell}_{R^α} has a unique maximal \mathcal{R} -augmentation $\mathbf{wtt}_x^- \rightarrow \mathbf{ok}_{x||\alpha}^+$;
- \mathbf{cell}_{W^α} has a unique maximal \mathcal{R} -augmentation $\mathbf{r}_y^- \rightarrow \mathbf{tt}_{\alpha||y}^+$;
- $\mathbf{cell}_{R^\alpha W^\beta}$ and $\mathbf{cell}_{W^\alpha R^\beta}$ are the empty \mathcal{R} -strategy.

A configuration will then be interpreted by the interaction between M and its memory store s :

$$\llbracket \langle M, s, \alpha \rangle \rrbracket = \langle M \rangle_\alpha \otimes \mathbf{cell}_s$$

Note that $(\llbracket M \rrbracket \otimes \mathbf{cell}_s) / \mathbf{q}_A^\alpha = \langle M \rangle_\alpha \otimes \mathbf{cell}_s$.

Residuation. If $s \leq_M s'$, then s' can be obtained from s using memory operations. There is a similar relation between cell strategies expressed via *residuation*:

Definition 9.1. Let $\sigma : A$ be a rigid \mathcal{R} -strategy, let $\mathfrak{q} \in \sigma$ and $\rho \in \mathcal{R}^{[\mathfrak{q}]^-}$, then, for $\mathfrak{q}' \in \sigma$ s.t. $\mathfrak{q} \hookrightarrow \mathfrak{q}' \in \sigma$, the *residual \mathcal{R} -augmentation of \mathfrak{q}' after (\mathfrak{q}, ρ)* is

$$\mathfrak{q}' / (\mathfrak{q}, \rho) = (\mathfrak{q}'_{\downarrow|\mathfrak{q}'|-|\mathfrak{q}|}, \lambda_{\mathfrak{q}' / (\mathfrak{q}, \rho)})$$

with \downarrow the projection on augmentations given in definition 4.7 and for every $a \in |\mathfrak{Q}'/(\mathfrak{Q}, \rho)|^+$, $\rho' \in \mathcal{R}^{[a]_{\mathfrak{Q}'/(\mathfrak{Q}, \rho)}^-}$,

$$\lambda(a^+)(\rho') = \lambda_{\mathfrak{Q}'}(a)(\rho_{\upharpoonright_{[a]_{\mathfrak{Q}}^-}} \cup \rho')$$

The *residual strategy of σ after (\mathfrak{Q}, ρ)* is then obtained componentwise by

$$\sigma/(\mathfrak{Q}, \rho) : A \downarrow (A - |\mathfrak{Q}|) = \{\mathfrak{Q}'/(\mathfrak{Q}, \rho) \mid \mathfrak{Q} \hookrightarrow \mathfrak{Q}' \in \sigma\}.$$

Note that the \mathcal{R} -strategy $(M)_\alpha$ is a particular case of residuation where $\sigma = \llbracket M \rrbracket$, $\mathfrak{Q} = \mathfrak{Q}_{A,x}^-$ and $\rho(\mathfrak{Q}_{A,x}^+) = \alpha$. We have compatibility with the cartesian monoidal categorical structure:

Proposition 9.1. *Let $\sigma : A \rightarrow C$ and $\tau : B \rightarrow D$ be two rigid \mathcal{R} -strategies, let $\mathfrak{Q} \in \sigma$, $\mathfrak{Q}' \in \tau$, $\rho \in \mathcal{R}^{|\mathfrak{Q}|^-}$ and $\rho' \in \mathcal{R}^{|\mathfrak{Q}'|^-}$, then*

$$\sigma/(\mathfrak{Q}, \rho) \otimes \tau/(\mathfrak{Q}', \rho') = (\sigma \otimes \tau)/(\mathfrak{Q} \otimes \mathfrak{Q}', \rho \otimes \rho')$$

for $\rho \otimes \rho'$ the union of ρ and ρ' with appropriate domain renaming. And similarly (for ρ, ρ' with appropriate domain renaming):

$$\begin{aligned} \langle \sigma/(\mathfrak{Q}, \rho), \tau \rangle &= \langle \sigma, \tau \rangle / (\{0\} \times \mathfrak{Q}, \rho) \\ \langle \sigma, \tau/(\mathfrak{Q}', \rho') \rangle &= \langle \sigma, \tau \rangle / (\{1\} \times \mathfrak{Q}', \rho') \end{aligned}$$

Proof. This is a direct check from the definitions, componentwise on the \mathcal{R} -augmentations in the strategies. \square

Proposition 9.2. *Let $\sigma : A \rightarrow B$ and $\tau : B \rightarrow C$ be rigid \mathcal{R} -strategies, let $\mathfrak{Q} \in \sigma$, $\mathfrak{Q}' \in \tau$, $\rho \in \mathcal{R}^{|\mathfrak{Q}|^-}$ and $\rho' \in \mathcal{R}^{|\mathfrak{Q}'|^-}$ be such that $\mathfrak{Q}, \mathfrak{Q}'$ are causally compatible and such that for every $e \in |\mathfrak{Q}|^-$, $e' \in |\mathfrak{Q}'|^-$ (writing $[e]_{\mathfrak{Q} \otimes \mathfrak{Q}'}^-$ for the set of predecessors of e in $\mathfrak{Q}' \otimes \mathfrak{Q}$ mapping to $(A^\perp \parallel C)^-$) we have $\rho(e) = \lambda_{\mathfrak{Q}' \otimes \mathfrak{Q}}(e)(\rho \cup \rho')_{\upharpoonright_{[e]_{\mathfrak{Q}' \otimes \mathfrak{Q}}^-}}$ and $\rho'(e') = \lambda_{\mathfrak{Q}' \otimes \mathfrak{Q}}(e')(\rho \cup \rho')_{\upharpoonright_{[e']_{\mathfrak{Q}' \otimes \mathfrak{Q}}^-}}$, then*

$$(\tau/(\mathfrak{Q}', \rho')) \odot (\sigma/(\mathfrak{Q}, \rho)) = (\tau \odot \sigma)/(\mathfrak{Q}' \odot \mathfrak{Q}, (\rho \cup \rho')_{\upharpoonright_{A^\perp \parallel C}})$$

Proof. Let us shorten residuations $\mathfrak{Q}'/(\mathfrak{Q}, \rho)$ into $\mathfrak{Q}'/\mathfrak{Q}$ and write \uplus to emphasize on the *disjoint* union of two sets.

First note that for $\mathfrak{Q}_\sigma \in \sigma$, $\mathfrak{Q}_\tau \in \tau$ such that $\mathfrak{Q} \hookrightarrow \mathfrak{Q}_\sigma$ and $\mathfrak{Q}' \hookrightarrow \mathfrak{Q}_\tau$, \mathfrak{Q}_σ and \mathfrak{Q}_τ are causally compatible iff $\mathfrak{Q}_\sigma/\mathfrak{Q}$ and $\mathfrak{Q}_\tau/\mathfrak{Q}'$ are. Indeed, let $\mathfrak{Q} = x_A \parallel x_B$, $\mathfrak{Q}' = x_B \parallel x_C$, $\mathfrak{Q}_\sigma = (x_A \uplus x'_A) \parallel (x_B \uplus x'_B)$ and $\mathfrak{Q}_\tau = (x_B \uplus x''_B) \parallel (x_C \uplus x'_C)$. Then, \mathfrak{Q}_σ agrees with \mathfrak{Q}_τ on B iff $x'_B = x''_B$ that is iff $\mathfrak{Q}_\sigma/\mathfrak{Q}$ and $\mathfrak{Q}_\tau/\mathfrak{Q}'$ agree on B as well. Furthermore,

$$\begin{aligned} (\leq_{\mathfrak{Q}_\sigma} \cup \leq_{\mathfrak{Q}_\tau})^* &= ((\leq_{\mathfrak{Q}_\sigma/\mathfrak{Q}} \uplus \leq_{\mathfrak{Q}}) \cup (\leq_{\mathfrak{Q}_\tau/\mathfrak{Q}'} \uplus \leq_{\mathfrak{Q}'}))^* \\ &= ((\leq_{\mathfrak{Q}_\sigma/\mathfrak{Q}} \cup \leq_{\mathfrak{Q}_\tau/\mathfrak{Q}'} \uplus (\leq_{\mathfrak{Q}} \cup \leq_{\mathfrak{Q}'}))^* \end{aligned}$$

So, by down-closure of \mathfrak{q} and \mathfrak{q}' in \mathfrak{q}_σ and \mathfrak{q}_τ respectively, $(\leq_{\mathfrak{q}_\sigma} \cup \leq_{\mathfrak{q}_\tau})^*$ defines a partial order iff $(\leq_{\mathfrak{q}_\sigma/\mathfrak{q}} \cup \leq_{\mathfrak{q}_\tau/\mathfrak{q}'})^*$ does (as there are no causal dependencies from events in $|\mathfrak{q}_\sigma/\mathfrak{q}|$ to events in $|\mathfrak{q}|$ and similarly for $|\mathfrak{q}_\tau/\mathfrak{q}'|$ and $|\mathfrak{q}'|$).

From the above, it is also immediate to see that $(\mathfrak{q}_\tau \otimes \mathfrak{q}_\sigma)/(\mathfrak{q}' \otimes \mathfrak{q}) = (\mathfrak{q}_\tau/\mathfrak{q}') \otimes (\mathfrak{q}_\sigma/\mathfrak{q})$ as partial order. Let us write \mathbb{Q} for this common order and compare the annotations from $(\mathfrak{q}_\tau \otimes \mathfrak{q}_\sigma)/(\mathfrak{q}' \otimes \mathfrak{q})$ and $(\mathfrak{q}_\tau/\mathfrak{q}') \otimes (\mathfrak{q}_\sigma/\mathfrak{q})$ inductively on $<_{\mathbb{Q}}$. *W.l.o.g* we study the case where $e \in ((A^\perp \parallel B) - |\mathfrak{q}|)$, the other case is symmetric:

$$\begin{aligned}
& \lambda_{(\mathfrak{q}_\tau \otimes \mathfrak{q}_\sigma)/(\mathfrak{q}' \otimes \mathfrak{q})}(e)(\nu) \\
&= \lambda_{\mathfrak{q}_\tau \otimes \mathfrak{q}_\sigma}(e) \left((\rho \cup \rho')_{\upharpoonright_{[e]_{\mathfrak{q}_\tau \otimes \mathfrak{q}_\sigma}^-}} \cup \nu \right) && \text{def. residuation} \\
&= \lambda_{\mathfrak{q}_\sigma}(e) \left\langle \lambda_{\mathfrak{q}_\tau \otimes \mathfrak{q}_\sigma}(e') \left((\rho \cup \rho')_{\upharpoonright_{[e]_{\mathfrak{q}_\tau \otimes \mathfrak{q}_\sigma}^-}} \cup \nu \right)_{\upharpoonright_{[e]_{\mathfrak{q}_\tau \otimes \mathfrak{q}_\sigma}^-}} \right\rangle_{e' \in [e]_{\mathfrak{q}_\sigma}^-} && \text{def. interaction} \\
&= \lambda_{\mathfrak{q}_\sigma}(e) \left\langle \lambda_{\mathfrak{q}_\tau \otimes \mathfrak{q}_\sigma}(e') \left((\rho \cup \rho')_{\upharpoonright_{[e]_{\mathfrak{q}_\tau \otimes \mathfrak{q}_\sigma}^-}} \cup \nu_{\upharpoonright_{[e]_{\mathfrak{q}_\tau \otimes \mathfrak{q}_\sigma}^-}} \right) \right\rangle_{e' \in [e]_{\mathfrak{q}_\sigma}^-} && \text{distribution of } \upharpoonright \\
&= \lambda_{\mathfrak{q}_\sigma}(e) \left\langle \lambda_{\mathfrak{q}_\tau \otimes \mathfrak{q}_\sigma}(e') \left((\rho \cup \rho')_{\upharpoonright_{[e]_{\mathfrak{q}_\tau \otimes \mathfrak{q}_\sigma}^-}} \cup \nu_{\upharpoonright_{[e]_{\mathfrak{q}_\tau \otimes \mathfrak{q}_\sigma}^-}} \right) \right\rangle_{e' \in [e]_{\mathfrak{q}_\sigma/\mathfrak{q}}^-} && \text{split +} \\
&\quad \cup \left\langle \lambda_{\mathfrak{q}_\tau \otimes \mathfrak{q}_\sigma}(e') (\rho \cup \rho')_{\upharpoonright_{[e]_{\mathfrak{q}_\tau \otimes \mathfrak{q}_\sigma}^-}} \right\rangle_{e' \in [e]_{\mathfrak{q}}^-} && \text{dom}(\nu) \subseteq Q \\
&= \lambda_{\mathfrak{q}_\sigma}(e) \left\langle \lambda_{(\mathfrak{q}_\tau \otimes \mathfrak{q}_\sigma)/(\mathfrak{q}' \otimes \mathfrak{q})}(e') (\nu_{\upharpoonright_{[e]_{\mathfrak{q}_\tau \otimes \mathfrak{q}_\sigma}^-}}) \right\rangle_{e' \in [e]_{\mathfrak{q}_\sigma/\mathfrak{q}}^-} && \text{def. residuation} \\
&\quad \cup \left\langle \rho(e') \right\rangle_{e' \in [e]_{\mathfrak{q}}^-} && \text{hypothesis} \\
&= \lambda_{\mathfrak{q}_\sigma}(e) \left\langle \lambda_{(\mathfrak{q}_\tau/\mathfrak{q}') \otimes (\mathfrak{q}_\sigma/\mathfrak{q})}(e') (\nu_{\upharpoonright_{[e]_{\mathfrak{q}_\tau \otimes \mathfrak{q}_\sigma}^-}}) \right\rangle_{e' \in [e]_{\mathfrak{q}_\sigma/\mathfrak{q}}^-} && \text{induction} \\
&\quad \cup \rho_{\upharpoonright_{[e]_{\mathfrak{q}}^-}} \\
&= \lambda_{\mathfrak{q}_\sigma/\mathfrak{q}}(e) \left\langle \lambda_{(\mathfrak{q}_\tau/\mathfrak{q}') \otimes (\mathfrak{q}_\sigma/\mathfrak{q})}(e') (\nu_{\upharpoonright_{[e]_{\mathfrak{q}_\tau \otimes \mathfrak{q}_\sigma}^-}}) \right\rangle_{e' \in [e]_{\mathfrak{q}_\sigma/\mathfrak{q}}^-} && \text{def. residuation} \\
&= \lambda_{(\mathfrak{q}_\tau/\mathfrak{q}') \otimes (\mathfrak{q}_\sigma/\mathfrak{q})}(e)(\nu) && \text{def. interaction}
\end{aligned}$$

Hence the two augmentations do have the same \mathcal{R} -annotations. \square

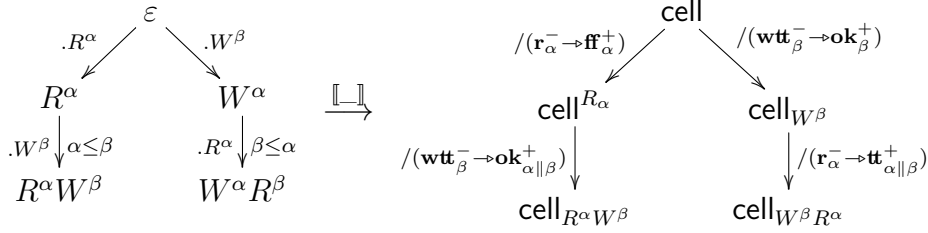
Finally, as a combination of projections and instantiations of annotation-functions preserving the order on \mathcal{R} , residuations preserve \leq_R on \mathcal{R} -strategies and can be decomposed:

Lemma 9.1. *Let $\sigma : A, \mathfrak{q} \in \sigma$ and two valuations $\rho, \rho' \in \mathcal{R}^{|\mathfrak{q}|^-}$, such that $\rho \leq_{\mathcal{R}} \rho'$, then $\sigma/(\mathfrak{q}, \rho) \leq_{\mathcal{R}} \sigma/(\mathfrak{q}, \rho')$.*

Lemma 9.2. *Let $\sigma : A, \mathfrak{q} \in \sigma$, and a valuation $\rho \in \mathcal{R}^{|\mathfrak{q}|^-}$, then, for every $\sigma \leq \sigma' : A$ with $\mathfrak{q} \leq_{\mathcal{R}} \mathfrak{q}' \in \sigma'$ we have $\sigma/(\mathfrak{q}, \rho) \leq_{\mathcal{R}} \sigma'/(\mathfrak{q}', \rho)$.*

Lemma 9.3. *Let $\sigma : A$ and $\mathfrak{q}_0 \hookrightarrow \mathfrak{q}_1 \in \sigma$ together with a valuation $\rho \in \mathcal{R}^{\mathfrak{q}_1^-}$, then $\sigma/(\mathfrak{q}_1, \rho) = (\sigma/(\mathfrak{q}_0, \rho_{\upharpoonright_{\mathfrak{q}_0}}))/(\mathfrak{q}_1/\mathfrak{q}_0, \rho_{\upharpoonright_{(\mathfrak{q}_1/\mathfrak{q}_0)}}$.*

Back to the cell strategies, we can now draw a correspondence between memory states and their corresponding strategies using the residual operation (with ρ directly specified on \mathfrak{q}):



So, for every $s \leq_{\mathcal{M}} s'$, there is a \mathcal{R} -augmentation $\mathfrak{q}_{s \triangleright s'} \in \text{cell}_s$ and a valuation $\rho_{s \triangleright s'}$ defined location-wise that matches the memory operation from s to s' and such that $\text{cell}_s / (\mathfrak{q}_{s \triangleright s'}, \rho_{s \triangleright s'}) = \text{cell}_{s'}$.

The interaction of a strategy with a memory store strategy is then formalised by:

Definition 9.2. Let $\sigma : \llbracket \Omega(s) \rrbracket^\perp \parallel (A)$ be a \mathcal{R} -strategy and $\mathfrak{q} \in \sigma$ be *fully compatible* with $\mathfrak{q}_{s \triangleright s'}$, that is $|\mathfrak{q}| = |\mathfrak{q}_{s \triangleright s'}|$, \mathfrak{q} and $\mathfrak{q}_{s \triangleright s'}$ are causally compatible and for every memory action $m \in \mathfrak{q}_{s \triangleright s'}^-$, $\lambda_{\mathfrak{q}}(m) = \rho_{s \triangleright s'}(m)$, then we write

$$\sigma / (\mathfrak{q} \otimes \mathfrak{q}_{s \triangleright s'}) : \llbracket \Omega(s') \rrbracket^\perp \parallel (A)$$

for the *residual of σ after* $(\mathfrak{q} \otimes \mathfrak{q}_{s \triangleright s'}, \lambda_{\mathfrak{q} \otimes \mathfrak{q}_{s \triangleright s'}} \upharpoonright \mathfrak{q}^-)$.

This is well defined as for every $e \in \mathfrak{q}_{s \triangleright s'}$, $\lambda_{\mathfrak{q} \otimes \mathfrak{q}_{s \triangleright s'}}(e) \in \mathcal{R}$. Informally, the definition above means that, considering some \mathfrak{q} which represents a scheduling of the memory operations turning s into s' , we extract from σ its behavior after the execution of these memory operations.

In the end our key lemma to prove theorem 9.1 will be:

Lemma 9.4. Let $\Omega(s_1) \vdash M : A$ and $\alpha \in \mathcal{R}$ define a valid configuration such that $\langle M, s_1, \alpha \rangle \Rightarrow \langle M', s'_1 \uplus s'_2, \alpha' \rangle$ with $\text{dom}(s_1) = \text{dom}(s'_1)$. Then, there exists $\mathfrak{q} \in (M)_\alpha$ fully compatible with $\mathfrak{q}_{s_1 \triangleright s'_1}$ such that:

$$(M)_\alpha / (\mathfrak{q} \otimes \mathfrak{q}_{s_1 \triangleright s'_1}) \leq_{\mathcal{R}} (M')_{\alpha'} \odot \text{cell}_{s'_2}$$

Instantiated with $\langle M, \emptyset, 0 \rangle \Rightarrow \langle \mathbf{skip}, s, \alpha \rangle$ this lemma yields soundness. The next two sections explain and prove it, by induction on the operational semantics. The critical cases are: assignment and dereferenciation exploiting that if $\alpha \leq_{\mathcal{R}} \beta$, then $\alpha \parallel \beta = \beta$ (which boils down to idempotence); and parallel composition, where compatibility of s' and s'' entails that the corresponding augmentations of cell_s are disjoint:

Lemma 9.5. *Let $s \leq_M s'$ and $s \leq_M s''$ be such that s' and s'' are compatible, i.e. $s' \uparrow s''$ is defined. Then, we also have $s \leq_Q s' \uparrow s''$ and*

$$(\mathfrak{q}_{s \triangleright (s' \uparrow s'')}, \rho_{s \triangleright s' \uparrow s''}) = (\mathfrak{q}_{s \triangleright s'} \uplus \mathfrak{q}_{s \triangleright s''}, \rho_{s \triangleright s'} \uplus \rho_{s \triangleright s''})$$

Proof. Straightforward by definition of $\llbracket s \rrbracket$ location by location. \square

9.1.2 Single step

We now go with the proof of lemma 9.4 in the case of one-step reductions. These cases are easy but they provide an incremental explanation for lemma 9.4. Unless stated otherwise, we assume that $\Omega(s_1) \vdash M : A$ and that $\langle M, s_1, \alpha \rangle$ defines a valid configuration.

Starting first with a standard substitution lemma:

Lemma 9.6. *Let $\Gamma, x : A \vdash M : B$ and $\Delta \vdash N : A$ then $\Gamma, \Delta \vdash M[N/x] : B$ and $\llbracket M[N/x] \rrbracket = \Gamma \otimes \Delta \xrightarrow{\Gamma \otimes \llbracket N \rrbracket} \Gamma \otimes A \xrightarrow{\llbracket M \rrbracket} B$.*

Proof. This is a simple consequence of the commutative monoidal structure of \otimes , the associativity of \odot and the naturality of Λ , **ev**, \langle, \rangle and **!**; noting that the inductive definition of $\llbracket - \rrbracket$ only uses these ingredients together with some base strategies for each computational features, and that M being an affine term, N (and its features) is only found once in $M[N/x]$. \square

For basic reductions that leave the store unchanged, our semantic then yields equality:

Lemma 9.7. *If $\langle M, s_1, \alpha \rangle \rightarrow \langle M', s_1, \alpha \rangle$ then $\llbracket M \rrbracket = \llbracket M' \rrbracket$.*

Proof. This is a direct verification on the semantics for the cases where M is **skip** $\parallel M_2$, $M_1 \parallel$ **skip**, **skip**; M_2 , **if tt** $M_1 M_2$, **if ff** $M_1 M_2$; and a consequence of the substitution lemma 9.6 together with monoidal closure for the case of $M = (\lambda x. M')N$.

For example, let us expand the case of $M_1 \parallel$ **skip**: by definition M_1 is a \mathcal{R} -strategy whose maximal configuration is either **run** $_x^-$ or **run** $_x^- \rightarrow$ **done** $_{f(x)}^+$ for some $f : \{x\} \rightarrow \mathcal{R}$. tensoring with $\llbracket \mathbf{skip} \rrbracket$ and post-composing with **par** yields a maximal configuration that is **run** $_x^-$ in the first case and **run** $_x^- \rightarrow$ **done** $_{0 \parallel f(x)}^+$ in the second case, which is indeed equal to $\llbracket M_1 \rrbracket$ as $0 \parallel f(x) = f(x)$. \square

Allowing for resource consumption we have:

Lemma 9.8. *If $\langle M, s_1, \alpha \rangle \rightarrow \langle M', s_1, \alpha' \rangle$ then $\langle M \rangle_\alpha = \langle M' \rangle_{\alpha'}$.*

Proof. The previous lemma induces that this is true on every cases where $\alpha = \alpha'$, and we need to check $\mathbf{consume}(\beta)$ separately: in that case $\alpha' = \alpha; \beta$ and by definition $(\mathbf{consume}(\beta))_\alpha$ has maximal configuration $\mathbf{done}_{\alpha; \beta}^+ = \mathbf{done}_{\alpha'}^+$ so is equal to $(\mathbf{skip})_{\alpha'}$. \square

Performing memory actions yields:

Lemma 9.9. *If $\langle M, s_1, \alpha \rangle \rightarrow \langle M', s'_1, \alpha' \rangle$ with $\text{dom}(s_1) = \text{dom}(s'_1)$ then there exists $\mathfrak{q} \in \langle M \rangle$ that is fully compatible with $\mathfrak{q}_{s_1 \triangleright s'_1}$ and such that $(M)_\alpha / (\mathfrak{q} \otimes \mathfrak{q}_{s_1 \triangleright s'_1}) = (M')_{\alpha'}$.*

Proof. The previous lemma induces that this is true on every cases where $s_1 = s'_1$, and we need to check the cases of $!\ell_R$ and $\ell_W := \mathbf{tt}$ (hence $\alpha = \alpha'$).

That \mathfrak{q} exists is obvious by definition of \mathbf{deref} and \mathbf{assign} . Equality on the \mathcal{R} -augmentations is direct too, relying, in the case where the memory action is not the first one to be performed on ℓ , on the fact that every annotation β in s_1 is less than α and so that the resulting annotation $\alpha \parallel \beta$ is actually equal to α , since $\alpha = \alpha \parallel 0 \leq_{\mathcal{R}} \alpha \parallel \beta \leq_{\mathcal{R}} \alpha \parallel \alpha = \alpha$. \square

Finally, reduction may uncover new memory locations. Those have to be hidden in order to preserve the equality between the interpretation of a term and the one of its reduct:

Lemma 9.10. *If $\langle M, s_1, \alpha \rangle \rightarrow \langle M', s'_1 \uplus s_2, \alpha' \rangle$ with $\text{dom}(s_1) = \text{dom}(s'_1)$ then there exists $\mathfrak{q} \in \langle M \rangle$ that is fully compatible with $\mathfrak{q}_{s_1 \triangleright s'_1}$ and such that $(M)_\alpha / (\mathfrak{q} \otimes \mathfrak{q}_{s_1 \triangleright s'_1}) = (M')_{\alpha'} \odot \mathbf{cell}_{s_2}$.*

Proof. Again the lemma 9.9 induces that this is true on every case where s_2 is empty. The remaining case is $M = \mathbf{new} x, y \mathbf{in} M_0$. In that case $\alpha = \alpha'$, $s_1 = s'_1$, $s_2 = [\ell \mapsto \varepsilon]$, and $M' = M_0[\ell_W/x, \ell_R/y]$, so, by substitution lemma (9.6), $(M_0)_\alpha = (M')_\alpha$ and, by compatibility of residuation with \odot (proposition 9.2): $(M)_\alpha = ([M] \odot \mathbf{cell}_\ell) / (\mathfrak{q}_{A, \alpha}^-) = (M')_\alpha \odot \mathbf{cell}_{s_2}$. \square

9.1.3 Many steps

We now move on to proving lemma 9.4 for the many-steps rules. From the previous section we show OTM; RFL is trivial. So far, we actually have equalities between the left and right hand-side. First we show that this is preserved if the reduction being considered *purely sequential*, meaning that its derivation tree does not contain any instance of the PAR rule (the contextual rule for parallel evaluation contexts are also excluded as they are particular cases of the PAR rule). This puts an emphasize on the fact that the operational semantics yields sub-optimal computation in comparison with the game semantics, only when it has to deal with parallel reductions.

Lemma 9.11. *If $\langle M, s_1, \alpha \rangle \Rightarrow \langle M', s'_1 \uplus s_2, \alpha' \rangle$ is a purely sequential reduction with $\text{dom}(s_1) = \text{dom}(s'_1)$ then there exists $\mathfrak{q} \in \llbracket M \rrbracket$ that is fully compatible with $\mathfrak{q}_{s_1 \triangleright s'_1}$ and such that*

$$\llbracket M \rrbracket_\alpha / (\mathfrak{q} \otimes \mathfrak{q}_{s_1 \triangleright s'_1}) = \llbracket M' \rrbracket_{\alpha'} \odot \text{cell}_{s_2}$$

Proof. By well-founded induction on the reduction tree, let us first study the contextual rules (without parallel contexts).

Consider $\langle M; N, s_1, \alpha \rangle \Rightarrow \langle M'; N, s'_1 \uplus s_2, \alpha' \rangle$ such that $\langle M, s_1, \alpha \rangle \Rightarrow \langle M', s'_1 \uplus s_2, \alpha' \rangle$. First note that by compatibility of residuations we have (i)

$$\begin{aligned} \llbracket M; N \rrbracket_\alpha &= (\text{seq} \odot (\llbracket M \rrbracket \otimes \llbracket N \rrbracket)) / (\text{run}_\alpha^- \rightarrow \text{run}_\alpha^+ \odot \text{run}_\alpha^-) \\ &= \text{seq} / (\text{run}_\alpha^- \rightarrow \text{run}_\alpha^+) \odot (\llbracket M \rrbracket \otimes \llbracket N \rrbracket) / \text{run}_\alpha^- \\ &= \text{seq} / (\text{run}_\alpha^- \rightarrow \text{run}_\alpha^+) \odot (\llbracket M \rrbracket_\alpha \otimes \llbracket N \rrbracket) \end{aligned}$$

and – also using the laws of \otimes and \odot – (ii)

$$\begin{aligned} \llbracket M'; N \rrbracket_{\alpha'} \odot \text{cell}_{s_2} &= (\text{seq} / (\text{run}_{\alpha'}^- \rightarrow \text{run}_{\alpha'}^+) \odot (\llbracket M \rrbracket_{\alpha'} \otimes \llbracket N \rrbracket)) \odot \text{cell}_{s_2} \\ &= \text{seq} / (\text{run}_{\alpha'}^- \rightarrow \text{run}_{\alpha'}^+) \odot ((\llbracket M \rrbracket_{\alpha'} \otimes \llbracket N \rrbracket) \odot \text{cell}_{s_2}) \\ &= \text{seq} / (\text{run}_{\alpha'}^- \rightarrow \text{run}_{\alpha'}^+) \odot ((\llbracket M \rrbracket_{\alpha'} \odot \text{cell}_{s_2}) \otimes \llbracket N \rrbracket) \end{aligned}$$

since M and N do not share variables.

Then, by induction there exists $\mathfrak{q} \in \llbracket M \rrbracket_\alpha$ fully compatible with $\mathfrak{q}_{s_1 \triangleright s'_1}$ such that $\llbracket M \rrbracket_\alpha / (\mathfrak{q} \otimes \mathfrak{q}_{s_1 \triangleright s'_1}) = \llbracket M' \rrbracket_{\alpha'} \odot \text{cell}_{s_2}$. By composition, $\mathfrak{q} = (\text{run}_\alpha^+ \odot \text{run}_\alpha^- \rightarrow \mathfrak{q})$ so \mathfrak{q} is also in $\llbracket M; N \rrbracket_\alpha$. Then, using (i) and the compatibility properties of residuation, we have

$$\llbracket M; N \rrbracket_\alpha / (\mathfrak{q} \otimes \mathfrak{q}_{s_1 \triangleright s'_1}) = \text{seq} / (\text{run}_\alpha^- \rightarrow \text{run}_\alpha^+) \odot (\llbracket M \rrbracket_\alpha / (\mathfrak{q} \otimes \mathfrak{q}_{s_1 \triangleright s'_1}) \otimes \llbracket N \rrbracket)$$

by induction, the right component is equal to $\llbracket M' \rrbracket_{\alpha'} \odot \text{cell}_{s_2} \otimes \llbracket N \rrbracket$, and by definition the left component is also equal to $\text{seq} / (\text{run}_{\alpha'}^- \rightarrow \text{run}_{\alpha'}^+)$ so we can finally apply (ii) to reach the desired equality.

The contextual rules with other evaluation contexts $\mathcal{E}[] = []N \mid \text{if } [] M N \mid [] := \mathbf{tt} \mid ![]$ can be treated similarly.

Thus, let us focus on the TRS rule, having:

$$\langle M, s_1, \alpha \rangle \Rightarrow \langle M', s'_1 \uplus s_2, \alpha' \rangle \Rightarrow \langle M'', s''_1 \uplus s'_2 \uplus s_3, \alpha'' \rangle$$

By induction hypothesis on the second reduction, there exists $\mathfrak{q} \in \llbracket M' \rrbracket_{\alpha'}$ fully compatible with $\mathfrak{q}_{s'_1 \uplus s_2 \triangleright s''_1 \uplus s'_2} \in \text{cell}_{s'_1 \uplus s_2}$. This last augmentation can actually be split into $\mathfrak{q}_{s'_1 \triangleright s''_1} \otimes \mathfrak{q}_{s_2 \triangleright s'_2}$ by definition of $\text{cell}_{s'_1 \uplus s_2} = \text{cell}_{s'_1} \otimes \text{cell}_{s_2}$.

So, in particular, \mathfrak{q} is causally compatible with $\mathfrak{q}_{s_2 \triangleright s'_2}$ and their composition $\mathfrak{q} \odot \mathfrak{q}_{s_2 \triangleright s'_2} \in \langle M' \rangle_{\alpha'} \odot \mathbf{cell}_{s_2}$ is fully compatible with $\mathfrak{q}_{s'_1 \triangleright s''_1}$.

By induction hypothesis on the first reduction, there is $\mathfrak{q}' \in \langle M \rangle_{\alpha}$ fully compatible with $\mathfrak{q}_{s_1 \triangleright s'_1}$ such that $\langle M \rangle_{\alpha} / (\mathfrak{q}' \otimes \mathfrak{q}_{s_1 \triangleright s'_1}) = \langle M' \rangle_{\alpha'} \odot \mathbf{cell}_{s_2}$ hence $\mathfrak{q} \odot \mathfrak{q}_{s_2 \triangleright s'_2} \in \langle M \rangle_{\alpha} / (\mathfrak{q}' \otimes \mathfrak{q}_{s_1 \triangleright s'_1})$ and so, there exists $\mathfrak{q}'' \in \langle M \rangle_{\alpha}$ such that $\mathfrak{q}' \hookrightarrow \mathfrak{q}''$, and $\mathfrak{q}'' / (\mathfrak{q}' \otimes \mathfrak{q}_{s_1 \triangleright s'_1}) = \mathfrak{q} \odot \mathfrak{q}_{s_2 \triangleright s'_2}$. Moreover, since $\mathfrak{q}_{s_1 \triangleright s''_1} / \mathfrak{q}_{s_1 \triangleright s'_1} = \mathfrak{q}_{s'_1 \triangleright s''_1}$, we have \mathfrak{q}'' is fully compatible with $\mathfrak{q}_{s_1 \triangleright s''_1}$. Now, using splits of residuations and their compatibility with \odot we have:

$$\begin{aligned} \langle M \rangle_{\alpha} / (\mathfrak{q}'' \otimes \mathfrak{q}_{s_1 \triangleright s''_1}) &= (\langle M \rangle_{\alpha} / (\mathfrak{q}' \otimes \mathfrak{q}_{s_1 \triangleright s'_1})) / ((\mathfrak{q} \odot \mathfrak{q}_{s_2 \triangleright s'_2}) \otimes \mathfrak{q}_{s'_1 \triangleright s''_1}) \\ &= (\langle M' \rangle_{\alpha'} \odot \mathbf{cell}_{s_2}) / (\mathfrak{q} \odot \mathfrak{q}_{s_2 \triangleright s'_2}, \lambda_{\mathfrak{q} \otimes \mathfrak{q}_{s'_1 \uplus s_2 \triangleright s'_1 \uplus s'_2}} \cup \rho_{s_2 \triangleright s'_2}) \\ &= (\langle M' \rangle_{\alpha'} / (\mathfrak{q} \otimes \mathfrak{q}_{s'_1 \triangleright s''_1})) \odot (\mathbf{cell}_{s_2} / \mathfrak{q}_{s_2 \triangleright s'_2}) \end{aligned}$$

By definition, the right hand-side of the resulting composition is equal to $\mathbf{cell}_{s'_2}$. On the left hand-side, the induction properties on \mathfrak{q} also give $\langle M' \rangle_{\alpha'} / (\mathfrak{q} \otimes \mathfrak{q}_{s'_1 \uplus s_2 \triangleright s'_1 \uplus s'_2}) = \langle M'' \rangle_{\alpha''} \odot \mathbf{cell}_{s_3}$. Putting everything together this yields

$$\langle M \rangle_{\alpha} / (\mathfrak{q}'' \otimes \mathfrak{q}_{s_1 \triangleright s''_1}) = \langle M'' \rangle_{\alpha''} \odot \mathbf{cell}_{s_3} \odot \mathbf{cell}_{s'_2}$$

as desired. \square

All operations used in the proof above (*i.e.* tensor, composition and residuation) preserve the pre-order $\leq_{\mathcal{R}}$ on \mathcal{R} -strategies so it is of no harm to remove the hypothesis of pure sequentiality on the many-step reduction $\langle M, s_1, \alpha \rangle \Rightarrow \langle M', s'_1 \uplus s_2, \alpha' \rangle$ and to change the induction hypothesis from equality to $\leq_{\mathcal{R}}$ in order to treat the cases of the CTX and TRS rules in the inductive proof of lemma 9.4.

Similarly, all the single-step reductions are treated by the lemmas from subsection 9.1.2 as they are particular cases of the general lemma 9.4.

Thus, to finish to prove lemma 9.4, we are left with the inductive case for the PAR many-step rules, which we treat now:

End of the proof of lemma 9.4. Consider

$$\frac{\langle M, s_1, \alpha \rangle \Rightarrow \langle M', s'_1 \uplus s_2, \alpha' \rangle \quad \langle N, s_1, \alpha \rangle \Rightarrow \langle N', s''_1 \uplus s_3, \alpha'' \rangle}{\langle M \parallel N, s_1, \alpha \rangle \Rightarrow \langle M' \parallel N', (s'_1 \uparrow s''_1) \uplus s_2 \uplus s_3, \alpha' \parallel \alpha'' \rangle} \text{PAR}$$

Similarly to other contextual rules in the proof of lemma 9.11, we first note that (i):

$$\begin{aligned} \langle M \parallel N \rangle_{\alpha} &= (\text{par} \odot (\llbracket M \rrbracket \otimes \llbracket N \rrbracket)) / (\mathbf{run}_{\alpha}^{-} \rightarrow (\mathbf{run}_{\alpha}^{+} \parallel \mathbf{run}_{\alpha}^{+}) \odot \mathbf{run}_{\alpha}^{-} \parallel \mathbf{run}_{\alpha}^{-}) \\ &= (\text{par} / (\mathbf{run}_{\alpha}^{-} \rightarrow (\mathbf{run}_{\alpha}^{+} \parallel \mathbf{run}_{\alpha}^{+}))) \odot ((\llbracket M \rrbracket \otimes \llbracket N \rrbracket) / \mathbf{run}_{\alpha}^{-} \parallel \mathbf{run}_{\alpha}^{-}) \\ &= (\text{par} / (\mathbf{run}_{\alpha}^{-} \rightarrow (\mathbf{run}_{\alpha}^{+} \parallel \mathbf{run}_{\alpha}^{+}))) \odot (\langle M \rangle_{\alpha} \otimes \langle N \rangle_{\alpha}) \end{aligned}$$

and that – writing $(\alpha' \parallel \alpha'')$ for the residuation factor $(\mathbf{run}_{\alpha' \parallel \alpha''}^{-} \rightarrow$

$(\mathbf{run}_{\alpha' \parallel \alpha''}^+ \parallel \mathbf{run}_{\alpha' \parallel \alpha''}^+)$ on $\mathbf{par} - (ii)$:

$$\begin{aligned}
& \langle M' \parallel N' \rangle_{\alpha' \parallel \alpha''} \odot \mathbf{cell}_{s_2 \uplus s_3} \\
&= \mathbf{par}/(\alpha' \parallel \alpha'') \odot \left(\langle M \rangle_{\alpha' \parallel \alpha''} \otimes \langle N \rangle_{\alpha' \parallel \alpha''} \right) \odot (\mathbf{cell}_{s_2} \otimes \mathbf{cell}_{s_3}) \\
&= \mathbf{par}/(\alpha' \parallel \alpha'') \odot \left(\left(\langle M \rangle_{\alpha' \parallel \alpha''} \odot \mathbf{cell}_{s_2} \right) \otimes \left(\langle N \rangle_{\alpha' \parallel \alpha''} \odot \mathbf{cell}_{s_3} \right) \right) \\
&\geq_{\mathcal{R}} \mathbf{par}/(\alpha \parallel \alpha) \odot \left(\left(\langle M \rangle_{\alpha'} \odot \mathbf{cell}_{s_2} \right) \otimes \left(\langle N \rangle_{\alpha''} \odot \mathbf{cell}_{s_3} \right) \right)
\end{aligned}$$

Then, by induction hypothesis there exists $\mathfrak{q} \in \langle M \rangle_{\alpha}$ and $\mathfrak{q}' \in \langle N \rangle_{\alpha}$ fully compatible with respectively $\mathfrak{q}_{s_1 \triangleright s'_1}$ and $\mathfrak{q}_{s_1 \triangleright s'_1}$. So $\mathfrak{q} \otimes \mathfrak{q}'$ is in $\langle M \parallel N \rangle_{\alpha}$ and is fully compatible with $\mathfrak{q}_{s_1 \triangleright s'_1 \uparrow s''_1}$ by lemma 9.5. Moreover, following (i),

$$\begin{aligned}
& \langle M \parallel N \rangle_{\alpha} / (\mathfrak{q} \otimes \mathfrak{q}' \otimes \mathfrak{q}_{s_1 \triangleright s'_1 \uparrow s''_1}) \\
&= (\mathbf{par}/(\alpha \parallel \alpha) \odot (\langle M \rangle_{\alpha} \otimes \langle N \rangle_{\alpha})) / (\mathfrak{q} \otimes \mathfrak{q}' \otimes \mathfrak{q}_{s_1 \triangleright s'_1 \uparrow s''_1}) \\
&= \mathbf{par}/(\alpha \parallel \alpha) \odot \left((\langle M \rangle_{\alpha} \otimes \langle N \rangle_{\alpha}) / (\mathfrak{q} \otimes \mathfrak{q}' \otimes \mathfrak{q}_{s_1 \triangleright s'_1} \otimes \mathfrak{q}_{s_1 \triangleright s'_1}) \right) \\
&= \mathbf{par}/(\alpha \parallel \alpha) \odot \left(\langle M \rangle_{\alpha} / (\mathfrak{q} \otimes \mathfrak{q}_{s_1 \triangleright s'_1}) \otimes \langle N \rangle_{\alpha} / (\mathfrak{q}' \otimes \mathfrak{q}_{s_1 \triangleright s'_1}) \right)
\end{aligned}$$

Which is $\leq_{\mathcal{R}}$ to $(\langle M' \parallel N' \rangle_{\alpha' \parallel \alpha''} \odot \mathbf{cell}_{s_2 \uplus s_3})$ by induction hypothesis and following following (ii). \square

9.2 Adequacy for \mathbb{R}_+ -IPA

In the previous section, we showed that if $M \Downarrow^{\alpha}$ then its corresponding strategy is non empty and moreover, there exists $\alpha' \leq \alpha$ such that $\mathbf{done}^{\alpha'} \in \langle M \rangle_0$. While from [CC16] we know that the converse is also true for the first part (*i.e.* a non empty denotation implies a (may) converging term), the second part does not hold in our model; in general our model is not adequate.

Non-Adequacy. To see why, consider:

$$\vdash \mathbf{new} \ x, \left(\begin{array}{c} \mathbf{wait}(1); \\ x_W := \mathbf{tt}; \\ \mathbf{wait}(2) \end{array} \parallel \begin{array}{c} \mathbf{wait}(2); \\ !x_R; \\ \mathbf{wait}(1) \end{array} \right) : \mathbf{bool} \ \mathbf{in}$$

Our model predicts that this may evaluate to \mathbf{tt} in 3 seconds (see Figure 8.9) and to \mathbf{ff} in 4 seconds. However, the operational semantics can only evaluate it (both to \mathbf{tt} and \mathbf{ff}) in 4 seconds. Intuitively, the reason is that the causal shapes implicit in the reduction \Rightarrow are all series-parallel (generated with sequential and parallel composition), whereas the interaction in Figure 8.9 is not.

Our causal semantic approach yields a finer resource analysis than the one achieved by the parallel operational semantics. The operational semantics, rather than our model, is to blame for non-adequacy: indeed, we now show that for $\mathcal{R} = \mathbb{R}_+$ our model is adequate with respect to an operational semantics specialized for time.

\mathbb{R}_+ -IPA To improve the performance of \mathbb{R}^+ -IPA (*i.e.* \mathcal{R} -IPA on time), we extend its operational semantics with the following rule:

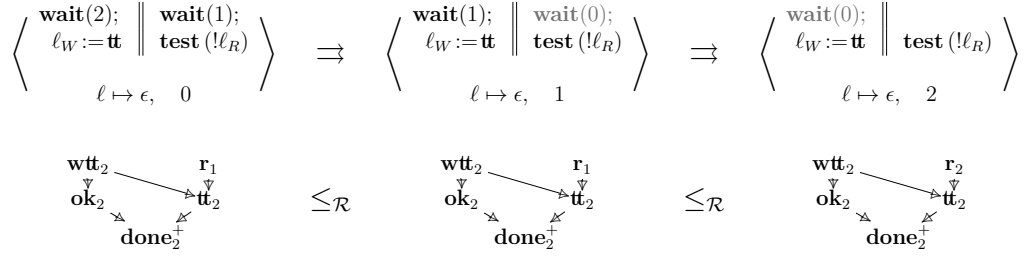
$$\langle \mathbf{wait}(t_1 + t_2), s, t_0 \rangle \rightarrow \langle \mathbf{wait}(t_2), s, t_0 + t_1 \rangle \quad (\text{split})$$

Also considering that $\mathbf{wait}(0) = \mathbf{skip}$, this rule allows for a parsimonious usage of resources: the scheduler might choose to stop a idling process and restart it later to improve the overall execution time. For example, the program above can now evaluate to \mathbf{tt} in 3 seconds.

Obviously, $(\mathbf{wait}(t_1 + t_2))_{t_0} = (\mathbf{wait}(t_2))_{t_0 + t_1}$, so soundness still holds for the finer operational semantics. Moreover, the new semantics is conservative with respect to the previous one and thus it can still yields sub-optimal results, so the statement of soundness remains as an inequality. Yet, we now show that for a valid configuration $\langle M, s, t \rangle$ with $\Omega(s) \vdash M : \mathbb{X}$, every augmentation $\mathfrak{q} \in (\mathbb{M})_t \otimes \mathbf{cell}_s$ that *returns at time* t_f (meaning that it has a top event $r_{t_f}^+$, its *result*, such that $r^+ \in (\mathbb{X})$ – *i.e.* $r_{t_f}^+$ is equal to either $\mathbf{done}_{t_f}^+$, $\mathbf{ok}_{t_f}^+$, $\mathbf{tt}_{t_f}^+$ or $\mathbf{ff}_{t_f}^+$), one can derive an optimal reduction leading $\langle M, s, t_0 \rangle$ to its corresponding *value* ($\mathbf{skip}, \ell, \mathbf{tt}, \mathbf{ff}$) at time t_f . This will prove adequacy.

Optimal reduction Intuitively, in the interpretation of a valid configuration $\langle M, s, t_0 \rangle$ with $\Omega(s) \vdash M : \mathbb{X}$, an augmentation $\mathfrak{q}_M \otimes \mathfrak{q}_s \in (\mathbb{M})_{t_0} \otimes \mathbf{cell}_s$ that results in time t_f describes an interaction between $(\mathbb{M})_{t_0}$ and the memory that yields a successful evaluation to its result at time t_f . To prove adequacy, we show that this evaluation can be reproduced at the level of terms by extracting a derivation from $\langle M, s, t_0 \rangle$ to its value at time t_f . This derivation will follow a simple pattern:

- performing every non-**wait** operation as soon as it is *available* in M (*i.e.* in head position) but *only if* it has been acknowledged/answered by the memory (the memory has its own schedule – described in \mathfrak{q}_s – and it may force to delay an available reduction);
- or waiting *optimally* until the first case is met, *i.e.* by reducing simultaneously all **waits** in head-position.

Figure 9.1: Spending time adequately (where $\text{test } M = \text{if } M \text{ skip } \perp$)

A representative example of the optimal reduction described above is depicted in figure 9.1 with the augmentation displayed below its configuration. The first step simply spends 1 second in parallel in the two threads, as none of them has an available non-**wait** operation. The second step does also spend time although $!l_R$ is available in M ; this is because the corresponding read has not yet been answered by the memory – witnessed by the fact that in the augmentation, \mathbf{tt} is causally dependent from \mathbf{wtt} whose corresponding write operation is not yet available in M . After this second, this write is made available in M so the third and fourth steps – not depicted in figure 9.1 – will thus simply be the sequential execution of the write and read operations.

Canonical form Apart from its top results \mathbf{ff}^+ , \mathbf{tt}^+ , \mathbf{ok}^+ or \mathbf{done}^+ , the arena associated with $\Omega(s) \vdash \mathbb{X}$ only allows to record memory operations in the denotation of a term, so we first discard cases where M has a non-waiting operation that is not a memory operation:

Definition 9.3. We say that a term M is *in canonical form* if it cannot be decomposed as $\mathcal{E}[(\lambda x.M) N]$, $\mathcal{E}[\text{skip}; N]$, $\mathcal{E}[\text{skip} \parallel N]$, $\mathcal{E}[N \parallel \text{skip}]$, $\mathcal{E}[\text{if } \mathbf{tt} N_1 N_2]$, $\mathcal{E}[\text{if } \mathbf{ff} N_1 N_2]$, or $\mathcal{E}[\text{new } x, y \text{ in } N]$ for $\mathcal{E}[-]$ an evaluation context.

Let $\mathfrak{q} \in (\mathbb{M})_{t_0} \otimes \text{cell}_s$, we say that it has a *minimal operation with timing* t if it is a result with time t or if it has a prefix $\mathfrak{p} \hookrightarrow \mathfrak{q}$ of the form $\mathfrak{p} = (\mathbf{wtt}_{\ell,t} \rightarrow \mathbf{ok}_{\ell,t})$ or $\mathfrak{p} = (\mathbf{r}_{\ell,t} \rightarrow b_{\ell,t})$ for $\ell \in \text{dom}(s)$ and $b \in \{\mathbf{tt}, \mathbf{ff}\}$. On figure 9.1, there is only one minimal operation $(\mathbf{wtt}_2 \rightarrow \mathbf{ok}_2)$ and it has timing 2.

We now relate minimal operations in \mathfrak{q} with available operations or values in M , when M is a ground term in canonical form. We show that if $t = t_0$, the corresponding operation can be performed immediately. Whereas if $t > t_0$ then we need to spend time to trigger it; it is then critical to spend time on *all the available waits* in parallel:

Lemma 9.12. *Let $\Omega(s) \vdash M : \mathbb{X}$ in canonical form and $t_0 \in \mathbb{R}_+$ describe a valid configuration, then for $\mathfrak{q} \in \langle M \rangle_{t_0} \otimes \text{cell}_s$ returning at time t_f we have – mutual exclusion:*

1. *either \mathfrak{q} has a minimal operation with timing t_0 and the corresponding value/operation is available in M (i.e. $M = \mathbf{skip}, \mathbf{ff}, \mathbf{tt}, \ell, \mathcal{E}[\ell]$ or $\mathcal{E}[\ell := \mathbf{tt}]$);*
2. *or all minimal operations have timing strictly greater than t_0 , and M can wait optimally, meaning that for $\alpha_{\min} = \min\{\alpha \mid M = \mathcal{E}[\mathbf{wait}(\alpha)]\}$, the minimal delay that can be found in an available \mathbf{wait} command in M , and every $\beta \leq \alpha_{\min}$ we have*

$$\langle M, s, t_0 \rangle \Rightarrow \langle M', s, t_0 + \beta \rangle$$

with M' only differing from M by having replaced the available $\mathbf{wait}(\alpha)$ commands in M with $\mathbf{wait}(\alpha - \alpha_{\min})$ commands. Furthermore $\mathfrak{q} \in \langle M' \rangle_{t_0 + \beta} \otimes \text{cell}_s$.

Proof. By induction on the structure of terms in canonical form:

If M is from an axiom of figure 8.1, it can either be a value ($\mathbf{skip}, \mathbf{tt}, \mathbf{ff}$, or $\ell \in \text{dom}(s)$), verifying 1; or it can be $\mathbf{wait}(\beta)$, verifying 2.

If $M = \mathbf{if} M' N_1 N_2$, then M' must be in canonical form and different from \mathbf{tt} or \mathbf{ff} . M also has the same available \mathbf{wait} commands as M' so they share the same α_{\min} .

Let $s = s_1 \uplus s_2$ such that $\Omega(s_1) \vdash M' : \mathbf{bool}$ and $\Omega(s_2) \vdash N_1, N_2 : \mathbb{X}$, then by definition of the interpretation of \mathbf{if} , if $\mathfrak{q} \in \langle M \rangle_{t_0} \otimes \text{cell}_s$ with result in time t_f , then there exists $\mathfrak{q}' \in \langle M' \rangle_{t_0} \otimes \text{cell}_{s_1}$ resulting in time $t_{f'}$ such that $(\mathfrak{q}' - \{b_{t_{f'}}^+\}) \hookrightarrow \mathfrak{q}$ and, if $b = \mathbf{tt}$ (respectively $b = \mathbf{ff}$) there exists $\mathfrak{q}_i \in \langle N_i \rangle_{t_{f'}} \otimes \text{cell}_{s_2}$ (for $i = 1$ or 2 respectively) such that \mathfrak{q} is the *concatenation of* $(\mathfrak{q}' - \{b_{t_{f'}}^+\})$ *with* \mathfrak{q}_i , that is the augmentation resulting in glueing $(\mathfrak{q}' - \{b_{t_{f'}}^+\})$ and \mathfrak{q}_i together by adding immediate causal links from the maximal events of $(\mathfrak{q}' - \{b_{t_{f'}}^+\})$ to the minimal event of \mathfrak{q}_i . In particular, this means that \mathfrak{q} has the same minimal operations as \mathfrak{q}' .

One can thus apply the induction hypothesis on M' to deduce that if \mathfrak{q} has a minimal operation with timing t_0 then so does \mathfrak{q}' , hence M' validates 1 and so M validates 1 as well.

On the other hand, if all minimal operations in \mathfrak{q} have timing greater than t_0 then so does the minimal operations in \mathfrak{q}' and so by induction hypothesis again M' validates 2, leading to M validating 2 as well: for every $\beta \leq \alpha_{\min}$, $\langle M', s_1, t_0 \rangle \Rightarrow \langle M'', s_1, t_0 + \beta \rangle$ yields $\langle M, s, t_0 \rangle \Rightarrow \langle \mathbf{if} M'' N_1 N_2, s, t_0 + \beta \rangle$ by applying the contextual rule of \Rightarrow (changing s_1 to s transparently); moreover,

the term $\mathbf{if} M'' N_1 N_2$ has the same available **wait** commands as M'' so by induction hypothesis it validates the condition of 2 and, similarly to the above deconstruction of the augmentations in $\langle M \rangle_{t_0} \otimes \mathbf{cell}_s$, one can deconstruct the augmentations in $\langle \mathbf{if} M'' N_1 N_2 \rangle_{t_0+\beta} \otimes \mathbf{cell}_s$ and see that $\mathfrak{q}' \in \langle M'' \rangle_{t_0+\beta} \otimes \mathbf{cell}_{s_1}$ yields $\mathfrak{q} \in \langle \mathbf{if} M'' N_1 N_2 \rangle_{t_0+\beta} \otimes \mathbf{cell}_s$.

The same reasoning holds for $M = M'; N \mid M' := \mathbf{wtt} \mid !M'$. The case of $M = M_1 \parallel M_2$ is also similar to the above **if** case, except that when \mathfrak{q} has all its minimal events with timing greater than t_0 then $\alpha_{\min} = \min(\alpha_{\min 1}, \alpha_{\min 2})$ and one has to reduce *both* M_1 and M_2 in parallel with time increased by the same $\beta \leq \alpha_{\min}$.

Finally, one can note that **new** x, y in M' is not a canonical form and, similarly, a simple induction on typing trees shows that $M' N$ cannot be in canonical form either (a term M' of type $A \multimap B$ can only be obtained from an abstraction or from the application of an abstraction). This concludes our induction. \square

Adequacy Alternating between optimal waiting phases and optimally scheduled non-**wait** operations, we can finally prove adequacy.

Theorem 9.2. *Let $\Omega(s) \vdash M : \mathbf{com}$, $t_0 \in \mathbb{R}_+$ describing a valid configuration, if $\mathfrak{q} \in \langle M \rangle_{t_0} \otimes \mathbf{cell}_s$ with maximal event $\mathbf{done}_{t_f}^+$. Then, there is a derivation*

$$\langle M, s, t_0 \rangle \Rightarrow \langle \mathbf{skip}, -, t_f \rangle$$

In particular, if $\vdash M : \mathbf{com}$ and $\mathbf{done}_{t_f}^+ \in \langle M \rangle_0$, then $M \Downarrow_{t_f}$.

Proof. By induction on the size of M . First, if M is not in canonical form, it has the form of one of the evaluation contexts from definition 9.3 so we can perform the corresponding one-step reduction: $\langle M, s, t_0 \rangle \rightarrow \langle M', s \uplus s', t_0 \rangle$ with $s'(\ell) = \varepsilon$ for every $\ell \in \text{dom}(s')$. Moreover, by soundness, $\langle M \rangle_{t_0} \otimes \mathbf{cell}_s = (\langle M' \rangle_{t_0} \odot \mathbf{cell}_{s'}) \otimes \mathbf{cell}_s$ and so \mathfrak{q} is of the form $(\mathfrak{q}_{\langle M' \rangle} \odot \mathfrak{q}_{s'}) \otimes \mathfrak{q}_s$. This yields $(\mathfrak{q}_{\langle M' \rangle} \otimes \mathfrak{q}_{s'}) \otimes \mathfrak{q}_s \in \langle M' \rangle_{t_0} \otimes \mathbf{cell}_{s \uplus s'}$ with maximal event $\mathbf{done}_{t_f}^+$ and M' smaller than M so by induction hypothesis: $\langle M, s, t_0 \rangle \Rightarrow \langle M', s \uplus s', t_0 \rangle \Rightarrow \langle \mathbf{skip}, -, t_f \rangle$

Now, if M is in canonical form and $\mathfrak{q} \in \langle M \rangle_{t_0}$ has all its minimal operations with timing strictly greater than t_0 , then applying lemma 9.12 we can conclude by induction hypothesis.

Otherwise, at least one minimal operation has timing t_0 . If it is a result, then we are done as this implies $M = \mathbf{skip}$. If it is a memory operation, assume that it is $(\mathbf{wtt}_{t_0} \rightarrow \mathbf{ok}_{t_0})$ and write $s' = s[\ell \mapsto s(\ell).W^{t_0}]$. Then by lemma 9.12 again, it follows that $M = \mathcal{E}[\ell_W := \mathbf{tt}]$ and $\mathfrak{q}/((\mathbf{wtt}_{t_0} \rightarrow \mathbf{ok}_{t_0})) \in$

$(\mathcal{E}[\mathbf{skip}])_{t_0} \otimes \text{cell}_{s'}$. So applying the induction hypothesis we get

$$\langle M, s, t_0 \rangle \Rightarrow \langle \mathcal{E}[\mathbf{skip}], s', t_0 \rangle \Rightarrow \langle \mathbf{skip}, -, t_f \rangle$$

The cases of $(\mathbf{r}_{t_0} \rightarrow \mathbf{tt}_{t_0})$, $(\mathbf{r}_{t_0} \rightarrow \mathbf{ff}_{t_0})$ are similar; concluding the induction. \square

9.3 Improvement for \mathbb{R}_+ -IPA

In the context of programs manipulations like compilation or optimization, formal semantics are used to prove that a program and its updated version behave the same. Usually, for qualitative aspect one is interested in *contextual equivalence* that ensures that two open terms behave similarly whatever their execution environment. More precisely, a *context* for a type $\Gamma \vdash A$ is defined as “a closed term with a hole” that is $C[-]$ of the form

$$\begin{aligned} C[-] ::= & [-] \mid C[-] N \mid M C[-] \mid C[-]; N \mid M; C[-] \\ & \mid C[-] := \mathbf{tt} \mid !C[-] \mid (C[-] \parallel N) \mid (M \parallel C[-]) \\ & \mid \mathbf{if} C[-] N_1 N_2 \mid \mathbf{if} M C[-] N_2 \mid \mathbf{if} M N_1 C[-] \end{aligned}$$

such that for every term $\Gamma \vdash M : A$, $\vdash C[M] : \mathbf{com}$; and two terms $\Gamma \vdash M, M' : A$ are said to be *contextually equivalent*, written $M \cong M'$, if for every context $C[-]$, $C[M] \Downarrow$ iff $C[M'] \Downarrow$ – for \Downarrow a given notion of convergence *e.g.* may in IPA.

In the case of quantitative operational semantics like \mathbb{R}_+ -IPA however, one may rather be interested in a notion of *improvement* between programs [San91]:

Definition 9.4 (Improvement). Let $\Gamma \vdash M, N : A$ be two terms of \mathbb{R}_+ -IPA, we say that M *may be improved by* N , written $M \gtrsim N$, if for every context $C[-]$, $C[M] \Downarrow^\alpha$ implies $C[N] \Downarrow^{\alpha'}$ for $\alpha' \leq_{\mathcal{R}} \alpha$.

In the above M and N may not be contextually equivalent: N may terminate more often and with better time than M . It is also common to strengthen the definition of improvement and ask:

Definition 9.5 (Strong improvement). Let $\Gamma \vdash M, N : A$ be two terms of \mathbb{R}_+ -IPA, we say that M is *strongly improved by* N , written $M \triangleright N$, if $M \gtrsim N$ and $M \cong N$.

Based on the soundness and adequacy results for \mathbb{R}^+ -IPA, we show in this section how the operational semantic of improvement reflect in our model. This is widely inspired by the *semantic of improvement* in [Ghi05].

9.3.1 Semantics of Improvement

Let ∞ be greater than any real number, given a closed term $\vdash C : \mathbf{com}$, we set

$$\mathbf{t}_{\min}(C) = \min \left(\{\infty\} \cup \{t \in \mathbb{R}_+ \mid C \Downarrow^t\} \right)$$

this corresponds to the *minimal* time of convergence of C or infinity if C diverges. The condition for improvement can then be rewritten as $\mathbf{t}_{\min}(C[M]) \geq \mathbf{t}_{\min}(C[N])$.

Based on soundness and adequacy, we have the following equivalence between \mathbb{R}_+ -IPA terms and their denotations:

Corollary 9.1. *Let $\vdash M : \mathbf{com}$ be a command of \mathbb{R}_+ -IPA, then*

$$\mathbf{t}_{\min}(M) = \mathbf{t}_{\min}(\llbracket M \rrbracket)$$

for $\mathbf{t}_{\min}(\sigma : \mathbf{com}) = \min \left(\{\infty\} \cup \{t \in \mathbb{R}_+ \mid \mathbf{done}^t \in \sigma / (\mathbf{run}_0^-)\} \right)$.

Following [Ghi05], the relations of improvement on terms then have a semantic counterpart:

Definition 9.6 (Semantics of improvement). Let $\sigma, \tau : A$ be two \mathcal{R} -strategies, we say that σ *improves* τ , written $\tau \gtrsim \sigma$, if for every \mathcal{R} -strategy $\rho : A \rightarrow \mathbf{com}$, $\mathbf{t}_{\min}(\rho \odot \tau) \geq \mathbf{t}_{\min}(\rho \odot \sigma)$.

If furthermore $\sigma = \tau$ as plain rigid strategies then σ is said to *strongly improve* τ , written $\tau \triangleright \sigma$.

Proposition 9.3 (Soundness of improvement). *Let $\Gamma \vdash M, N : A$, then*

$$\llbracket M \rrbracket \gtrsim \llbracket N \rrbracket \implies M \gtrsim N$$

$$\llbracket M \rrbracket \triangleright \llbracket N \rrbracket \implies M \triangleright N$$

Proof. Let $C[-]$ be a context for M and N , then $C[-]$ defines a \mathbb{R}_+ -strategy $\llbracket C \rrbracket : \Gamma^\perp \parallel A \rightarrow \mathbf{com}$ such that $\llbracket C[M] \rrbracket = \llbracket C \rrbracket \odot \llbracket M \rrbracket$ and $\llbracket C[N] \rrbracket = \llbracket C \rrbracket \odot \llbracket N \rrbracket$ so by hypothesis and corollary 9.1 $\mathbf{t}_{\min}(C[M]) = \mathbf{t}_{\min}(\llbracket C[M] \rrbracket) \geq \mathbf{t}_{\min}(\llbracket C[N] \rrbracket) = \mathbf{t}_{\min}(C[N])$.

Moreover, if $\llbracket M \rrbracket = \llbracket N \rrbracket$ as plain strategies then $\llbracket C[M] \rrbracket = \llbracket C[N] \rrbracket$ as plain strategies as well so, by definition, $\mathbf{t}_{\min}(\llbracket C[M] \rrbracket) = \infty$ iff $\mathbf{t}_{\min}(\llbracket C[N] \rrbracket) = \infty$, and so, by corollary 9.1 again, $C[M]$ converges iff $C[N]$ converges. \square

The above relation allows to reason on denotations instead of terms in order to show improvements, yet, as they stand the relations of semantic improvement are not much handier to work with than their operational counterpart. However, they have a direct connection with the – more locally defined – pre-orders on \mathbb{R}_+ -strategies presented in section 8.2.3:

9.3.2 Examples of improvement

Despite being non-adequate, the relations $\leq_{\mathcal{R}}$ and $\preceq_{\mathcal{R}}$ can still be used to show some “laws of improvement” on \mathbb{R}^+ -IPA programs, in the style of the laws of improvement presented in [Ghi05]. For example, the following parallel optimisation is a strong improvement:

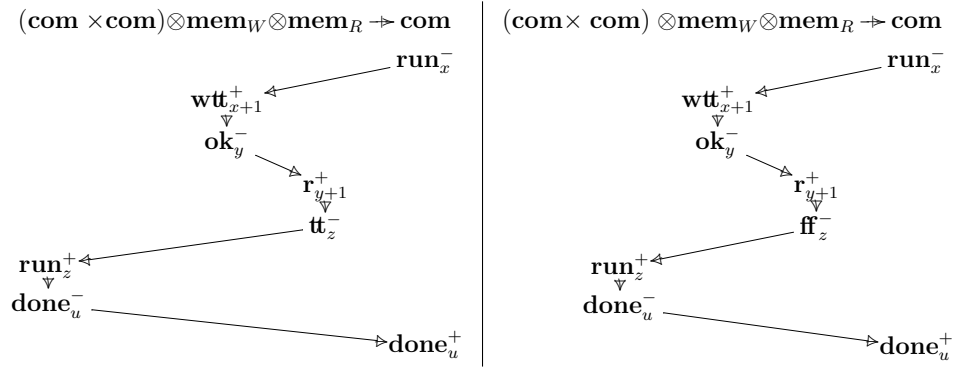
$$\mathbf{new\ } y \mathbf{ in} \quad M; (\mathbf{wait}(1); x_W := \mathbf{tt}); \mathbf{if} (\mathbf{wait}(1); !y_R) N_1 N_2 \quad (\text{opt1})$$

$$\supseteq \quad \mathbf{new\ } y \mathbf{ in} \quad M; \mathbf{if} (\mathbf{wait}(1); x_W := \mathbf{tt}) \parallel (\mathbf{wait}(1); !y_R) N_1 N_2$$

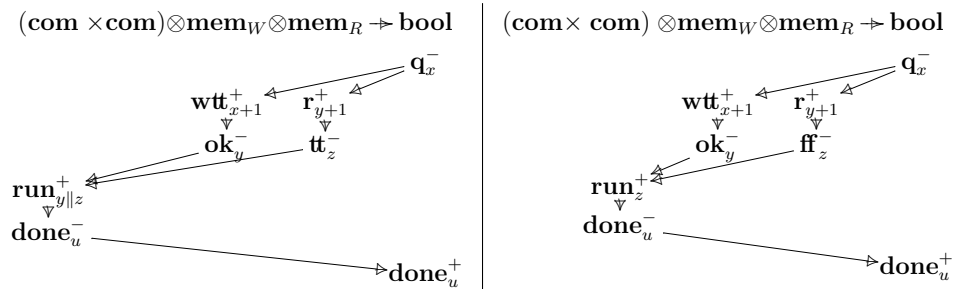
It means that in a context where performing memory operations takes 1 millisecond, then doing two independent memory operations in parallel is an improvement compare to executing them sequentially.

Lemma 9.13. *Let P_1 and P_2 be respectively the top and bottom terms in *opt1*, then $P_1 \supseteq P_2$.*

Proof. First let us compute the denotation of the open term $P'_1 = \mathbf{wait}(1); x_W := \mathbf{tt}; \mathbf{if} (\mathbf{wait}(1); !y_R)$, it yields a strategy with maximal augmentations:



Similarly, for $P'_2 = \mathbf{if} (\mathbf{wait}(1); x_W := \mathbf{tt}) \parallel (\mathbf{wait}(1); !y_R)$ we get:



Now, the whole interpretation of P_1 and P_2 is given by – for $i = 1, 2$:

$$(\text{seq} \odot (\llbracket M \rrbracket \otimes (\llbracket P'_i \rrbracket \odot \langle \llbracket N_1 \rrbracket, \llbracket N_2 \rrbracket \rangle))) \odot \text{cell}$$

and, by definition of **seq**, **cell** and associativity of composition, if M is writing on x , then this is equal to

$$\text{seq} \odot ((\llbracket M \rrbracket \odot (\mathbf{wtt}_x^- \rightarrow \mathbf{ok}_x^+)) \otimes ((\llbracket P'_i \rrbracket \odot (\mathbf{r}_x^- \rightarrow \mathbf{tt}_{t||x}^+)) \odot \langle \llbracket N_1 \rrbracket, \llbracket N_2 \rrbracket \rangle))$$

for some $t \in \mathbb{R}_+$, in which case we have

$$\llbracket P'_1 \rrbracket \odot (\mathbf{r}_x^- \rightarrow \mathbf{tt}_{t||x}^+) \geq_{\mathbb{R}_+} \llbracket P'_2 \rrbracket \odot (\mathbf{r}_x^- \rightarrow \mathbf{tt}_{t||x}^+)$$

and so by preservation of $\geq_{\mathbb{R}_+}$, $\llbracket P_1 \rrbracket \geq_{\mathbb{R}_+} \llbracket P_2 \rrbracket$, yielding $P_1 \sqsupseteq P_2$ by propositions 9.4 and 9.3. On the other hand, if M is not writing into the memory, then the denotations of P_1 and P_2 are equal to

$$\text{seq} \odot (\llbracket M \rrbracket \otimes ((\llbracket P'_i \rrbracket \odot (\mathbf{r}_x^- \rightarrow \mathbf{ff}_x^+)) \odot (\langle \llbracket N_1 \rrbracket, \llbracket N_2 \rrbracket \rangle \odot (\mathbf{wtt}_x^- \rightarrow \mathbf{ok}_{t||x}^+))))$$

in that case we still have

$$\llbracket P'_1 \rrbracket \odot (\mathbf{r}_x^- \rightarrow \mathbf{ff}_x^+) \geq_{\mathbb{R}_+} \llbracket P'_2 \rrbracket \odot (\mathbf{r}_x^- \rightarrow \mathbf{ff}_x^+)$$

concluding on the strong improvement of **opt1**. \square

Following the same reasoning we show that if instead y_R is bound with x_W then **opt1** is still a may improvement:

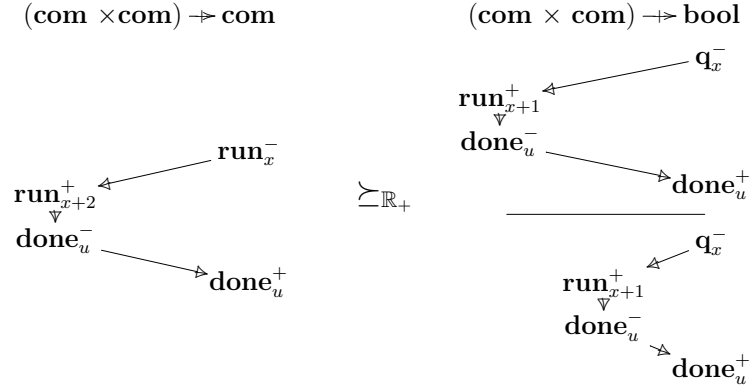
Lemma 9.14. *The following shuffling is a may improvement:*

$$\begin{aligned} & \mathbf{new } y \mathbf{ in} \\ & M; (\mathbf{wait}(1); y_W := \mathbf{tt}); \mathbf{if } (\mathbf{wait}(1); !y_R) N_1 N_2 \quad (\text{opt2}) \\ & \gtrsim \\ & \mathbf{new } y \mathbf{ in} \\ & M; \mathbf{if } (\mathbf{wait}(1); y_W := \mathbf{tt}) \parallel (\mathbf{wait}(1); !y_R) N_1 N_2 \end{aligned}$$

Proof. Let again write P_1 and P_2 for the top and bottom programs being compared. Then contrary to the above, their denotations are now equal to

$$\text{seq} \odot (\llbracket M \rrbracket \otimes ((\llbracket P'_i \rrbracket \odot \text{cell}) \odot \langle \llbracket N_1 \rrbracket, \llbracket N_2 \rrbracket \rangle))$$

It is not the case that $\llbracket P'_1 \rrbracket \odot \text{cell} \geq_{\mathbb{R}^+} \llbracket P'_2 \rrbracket \odot \text{cell}$ yet we still have $\llbracket P'_1 \rrbracket \odot \text{cell} \succeq_{\mathbb{R}^+} \llbracket P'_2 \rrbracket \odot \text{cell}$:



So, by preservation of $\succeq_{\mathbb{R}^+}$, $\llbracket P_1 \rrbracket \succeq_{\mathbb{R}^+} \llbracket P_2 \rrbracket$, yielding $P_1 \succsim P_2$ by proposition 9.4 and corollary 9.3. \square

9.3.3 Further improvements...

On \mathbb{R}^+ -IPA, it would be interesting to compare our model with structures used in timing analysis. For instance [MB12] relies on a concurrent generalization of control flow graphs that is reminiscent of event structures.

In an other line of work, it would also be worthy to adapt techniques known on plain concurrent games model such as symmetries [CCW15, CCW19] or essential events [CCHW18] to our annotated model. In the first case, this would allow us to deal with replication and languages with a fixpoint operator. In the second case, this would provide a setting to interpret *must convergence*, a desirable complement to the *may convergence* treated here, as it would give a semantics for *exact* improvements of non-deterministic programs. As suggested by recent experience with concurrent games [CCPW18, CdVW19], we expect these techniques to adapt transparently to annotations.

A more intricate question that is left open is whether our model could adapt to non-idempotent parallel usage of resources, such as power or bandwidth. Although we could predict their maximal usage for non-interfering programs (*i.e.* programs in which every memory cell is used in a sequential way only), it is not clear how to exploit information from possible interferences. Essential events might again be of help for such question.

Finally, a broader perspective for future work is to investigate whether our annotated construction could be used for other purposes than resources analysis, as *e.g.* for symbolic execution or abstract interpretation. This would

require annotations to affect somehow the execution flow of strategies, for which the more general concurrent games model with annotations sketched in section 3.3 might help.

Bibliography

- [ACHW18] Aurore Alcolei, Pierre Clairambault, Martin Hyland, and Glynn Winskel. The True Concurrency of Herbrand's Theorem. In Dan Ghica and Achim Jung, editors, *27th EACSL Annual Conference on Computer Science Logic (CSL 2018)*, volume 119 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 5:1–5:22, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.
- [ACL19] Aurore Alcolei, Pierre Clairambault, and Olivier Laurent. Resource-tracking concurrent games. In Mikołaj Bojańczyk and Alex Simpson, editors, *Foundations of Software Science and Computation Structures*, pages 27–44, Cham, 2019. Springer International Publishing.
- [AHM98] Samson Abramsky, Kohei Honda, and Guy McCusker. A fully abstract game semantics for general references. In *Proceedings. Thirteenth Annual IEEE Symposium on Logic in Computer Science (Cat. No. 98CB36226)*, pages 334–344. IEEE, 1998.
- [AJ94] Samson Abramsky and Radha Jagadeesan. Games and full completeness for multiplicative linear logic. *The Journal of Symbolic Logic*, 59(2):543–574, 1994.
- [AJM00] Samson Abramsky, Radha Jagadeesan, and Pasquale Malacaria. Full abstraction for PCF. *Inf. Comput.*, 163(2):409–470, 2000.
- [AM96] Samson Abramsky and Guy McCusker. Linearity, sharing and state: a fully abstract game semantics for idealized algol with active expressions. *Electr. Notes Theor. Comput. Sci.*, 3:2–14, 1996.
- [AM99] Samson Abramsky and Paul-André Melliès. Concurrent games and full completeness. In *14th Annual IEEE Symposium on Logic in Computer Science, Trento, Italy, July 2-5, 1999*, pages 431–442, 1999.
- [BB94] Franco Barbanera and Stefano Berardi. A symmetric lambda calculus for "classical" program extraction. In *Theoretical Aspects of*

Computer Software, International Conference TACS '94, Sendai, Japan, April 19-22, 1994, Proceedings, pages 495–515, 1994.

- [BDER97] Patrick Baillot, Vincent Danos, Thomas Ehrhard, and Laurent Regnier. Believe it or not, ajm’s games model is a model of classical linear logic. In *Proceedings of Twelfth Annual IEEE Symposium on Logic in Computer Science*, pages 68–75. IEEE, 1997.
- [BGMZ14] Aloïs Brunel, Marco Gaboardi, Damiano Mazza, and Steve Zdancewic. A core quantitative coefficient calculus. In *Programming Languages and Systems - 23rd European Symposium on Programming, ESOP 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014, Proceedings*, pages 351–370, 2014.
- [Bla92] Andreas Blass. A game semantics for linear logic. *Annals of Pure and Applied logic*, 56(1-3):183–220, 1992.
- [BN98] Franz Baader and Tobias Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.
- [Bus94] Samuel R Buss. On herbrand’s theorem. In *International Workshop on Logic and Computational Complexity*, pages 195–209. Springer, 1994.
- [Cas17] S. Castellan. *Structures concurrentes en sémantique des jeux, PhD thesis*. PhD thesis, University of Lyon, 2017.
- [CC16] Simon Castellan and Pierre Clairambault. Causality vs. interleavings in concurrent game semantics. In Josée Desharnais and Radha Jagadeesan, editors, *27th International Conference on Concurrency Theory, CONCUR 2016, August 23-26, 2016, Québec City, Canada*, volume 59 of *LIPICs*, pages 32:1–32:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.
- [CCHW18] Simon Castellan, Pierre Clairambault, Jonathan Hayman, and Glynn Winskel. Non-angelic concurrent game semantics. In *International Conference on Foundations of Software Science and Computation Structures*, pages 3–19. Springer, Cham, 2018.
- [CCPW18] Simon Castellan, Pierre Clairambault, Hugo Paquet, and Glynn Winskel. The concurrent game semantics of probabilistic PCF.

- In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, pages 215–224, 2018.
- [CCRW17] Simon Castellan, Pierre Clairambault, Silvain Rideau, and Glynn Winskel. Games and strategies as event structures. *Logical Methods in Computer Science*, 13(3), 2017.
- [CCW15] Simon Castellan, Pierre Clairambault, and Glynn Winskel. The parallel intensionally fully abstract games model of PCF. In *30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015, Kyoto, Japan, July 6-10, 2015*, pages 232–243, 2015.
- [CCW19] Simon Castellan, Pierre Clairambault, and Glynn Winskel. Thin games with symmetry and concurrent hybrid games. *To appear in Logical Methods in Computer Science.*, 2019.
- [CdVW19] Pierre Clairambault, Marc de Visme, and Glynn Winskel. Game semantics for quantum programming. *PACMPL*, 3(POPL):32:1–32:29, 2019.
- [Coq95] Thierry Coquand. A semantics of evidence for classical arithmetic. *J. Symb. Log.*, 60(1):325–337, 1995.
- [CP19] Simon Castellan and Hugo Paquet. Probabilistic programming inference via intensional semantics. In Luís Caires, editor, *Programming Languages and Systems*, pages 322–349, Cham, 2019. Springer International Publishing.
- [CS97] J.R.B. Cockett and R.A.G. Seely. Weakly distributive categories. *Journal of Pure and Applied Algebra*, 114(2):133–173, 1997.
- [DH02] Vincent Danos and Russell S Harmer. Probabilistic game semantics. *ACM Transactions on Computational Logic (TOCL)*, 3(3):359–382, 2002.
- [DJS97] Vincent Danos, Jean-Baptiste Joinet, and Harold Schellinx. A new deconstructive logic: Linear logic. *J. Symb. Log.*, 62(3):755–807, 1997.
- [Ehr12] Thomas Ehrhard. The Scott model of linear logic is the extensional collapse of its relational model. *Theor. Comput. Sci.*, 424:20–45, 2012.

- [FH07] Marcelo P. Fiore and Chung-Kil Hur. Equational systems and free constructions (extended abstract). In *Automata, Languages and Programming, 34th International Colloquium, ICALP 2007, Wroclaw, Poland, July 9-13, 2007, Proceedings*, pages 607–618, 2007.
- [FP07] Carsten Führmann and David J. Pym. On categorical models of classical logic and the geometry of interaction. *Mathematical Structures in Computer Science*, 17(5):957–1027, 2007.
- [FP09] Claudia Faggian and Mauro Piccolo. Partial orders, event structures and linear strategies. In *Typed Lambda Calculi and Applications, 9th International Conference, TLCA 2009, Brasilia, Brazil, July 1-3, 2009. Proceedings*, pages 95–111, 2009.
- [Gen35] Gerhard Gentzen. Untersuchungen über das logische schließen. i. *Mathematische zeitschrift*, 39(1):176–210, 1935.
- [Ghi05] Dan R. Ghica. Slot games: a quantitative model of computation. In *Proceedings of the 32nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2005, Long Beach, California, USA, January 12-14, 2005*, pages 85–97, 2005.
- [Gir87] Jean-Yves Girard. Linear logic. *Theor. Comput. Sci.*, 50:1–102, 1987.
- [Gir91] Jean-Yves Girard. A new constructive logic: Classical logic. *Mathematical Structures in Computer Science*, 1(3):255–296, 1991.
- [GK05] Philipp Gerhardy and Ulrich Kohlenbach. Extracting herbrand disjunctions by functional interpretation. *Arch. Math. Log.*, 44(5):633–644, 2005.
- [GM08] Dan R. Ghica and Andrzej S. Murawski. Angelic semantics of fine-grained concurrency. *Ann. Pure Appl. Logic*, 151(2-3):89–114, 2008.
- [Gog89] Joseph A. Goguen. What is unification? a categorical view of substitution, equation and solution. 1989.
- [GS14] Dan R. Ghica and Alex I. Smith. Bounded linear types in a resource semiring. In *Programming Languages and Systems -*

- 23rd European Symposium on Programming, ESOP 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014, Proceedings*, pages 331–350, 2014.
- [HDP93] Martin Hyland and Valeria De Paiva. Full intuitionistic linear logic. *Annals of Pure and Applied Logic*, 64(3):273–291, 1993.
- [Hei10] Willem Heijltjes. Classical proof forestry. *Ann. Pure Appl. Logic*, 161(11):1346–1366, 2010.
- [Her30] J. Herbrand. *Recherches sur la théorie de la démonstration*, PhD thesis, University of Paris, 1930.
- [HM99] Russell Harmer and Guy McCusker. A fully abstract game semantics for finite nondeterminism. In *Proceedings. 14th Symposium on Logic in Computer Science (Cat. No. PR00158)*, pages 422–430. IEEE, 1999.
- [HMSW11] Tony Hoare, Bernhard Möller, Georg Struth, and Ian Wehrman. Concurrent Kleene algebra and its foundations. *J. Log. Algebr. Program.*, 80(6):266–296, 2011.
- [HO00] J. M. E. Hyland and C.-H. Luke Ong. On full abstraction for PCF: I, II, and III. *Inf. Comput.*, 163(2):285–408, 2000.
- [HW13] Stefan Hetzl and Daniel Weller. Expansion trees with cut. *CoRR*, abs/1308.0428, 2013.
- [HY97] Kohei Honda and Nobuko Yoshida. Game theoretic analysis of call-by-value computation. In *International Colloquium on Automata, Languages, and Programming*, pages 225–236. Springer, 1997.
- [Koh99] Ulrich Kohlenbach. On the no-counterexample interpretation. *J. Symb. Log.*, 64(4):1491–1511, 1999.
- [Kri09] Jean-Louis Krivine. Realizability in classical logic. *Panoramas et synthèses*, 27:197–229, 2009.
- [Lai97] James Laird. Full abstraction for functional languages with control. In *Proceedings of Twelfth Annual IEEE Symposium on Logic in Computer Science*, pages 58–67. IEEE, 1997.

- [Lai01] James Laird. A fully abstract game semantics of local exceptions. In *Proceedings 16th Annual IEEE Symposium on Logic in Computer Science*, pages 105–114. IEEE, 2001.
- [Lai06] James Laird. Game semantics for higher-order concurrency. In *International Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 417–428. Springer, 2006.
- [Lau10] Olivier Laurent. Game semantics for first-order logic. *Logical Methods in Computer Science*, 6(4), 2010.
- [Law69] F William Lawvere. Adjointness in foundations. *Dialectica*, 23(3-4):281–296, 1969.
- [LMMP13] J. Laird, G. Manzonetto, G. McCusker, and M. Pagani. Weighted relational models of typed lambda-calculi. In *28th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), New Orleans, USA, Proceedings*, pages 301–310, 2013.
- [Lor60] Paul Lorenzen. Logik und agon. *Atti del Congresso Internazionale di Filosofia*, page 187–194, 1960.
- [MB12] Robert Mittermayr and Johann Blieberger. Timing analysis of concurrent programs. In Tullio Vardanega, editor, *12th International Workshop on Worst-Case Execution Time Analysis, WCET 2012, July 10, 2012, Pisa, Italy*, volume 23 of *OASICS*, pages 59–68. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012.
- [McK13] Richard McKinley. Proof nets for Herbrand’s theorem. *ACM Trans. Comput. Log.*, 14(1):5, 2013.
- [Mel05] Paul-André Melliès. Asynchronous games 4: A fully complete model of propositional linear logic. In *20th IEEE Symposium on Logic in Computer Science (LICS 2005), 26-29 June 2005, Chicago, IL, USA, Proceedings*, pages 386–395, 2005.
- [Mel12] Paul-André Melliès. Game semantics in string diagrams. In *Proceedings of the 27th Annual IEEE Symposium on Logic in Computer Science, LICS 2012, Dubrovnik, Croatia, June 25-28, 2012*, pages 481–490, 2012.

- [Mil87] Dale A Miller. A compact representation of proofs. *Studia Logica*, 46(4):347–370, 1987.
- [Mim11] Samuel Mimram. The structure of first-order causality. *Mathematical Structures in Computer Science*, 21(1):65–110, 2011.
- [MM07] Paul-André Melliès and Samuel Mimram. Asynchronous games: Innocence without alternation. In *CONCUR 2007 - Concurrency Theory, 18th International Conference, CONCUR 2007, Lisbon, Portugal, September 3-8, 2007, Proceedings*, pages 395–411, 2007.
- [Pan17] Prakash Panangaden. The 2017 alonzo church award. *ACM SIGLOG News*, 4(3):3–9, July 2017.
- [Par92] Michel Parigot. Lambda-my-calculus: An algorithmic interpretation of classical natural deduction. In *Logic Programming and Automated Reasoning, International Conference LPAR'92, St. Petersburg, Russia, July 15-20, 1992, Proceedings*, pages 190–201, 1992.
- [RW11] Silvain Rideau and Glynn Winskel. Concurrent strategies. In *Proceedings of the 26th Annual IEEE Symposium on Logic in Computer Science, LICS 2011, June 21-24, 2011, Toronto, Ontario, Canada*, pages 409–418, 2011.
- [San91] David Sands. Operational theories of improvement in functional languages (extended abstract). In *Functional Programming, Glasgow 1991, Proceedings of the 1991 Glasgow Workshop on Functional Programming, Portree, Isle of Skye, UK, 12-14 August 1991*, pages 298–311, 1991.
- [Sco82] Dana S Scott. Domains for denotational semantics. In *International Colloquium on Automata, Languages, and Programming*, pages 577–610. Springer, 1982.
- [See83] Robert A. G. Seely. Hyperdoctrines, natural deduction and the Beck condition. *Math. Log. Q.*, 29(10):505–542, 1983.
- [See87] Robert AG Seely. *Linear logic, *-autonomous categories and cofree coalgebras*. Ste. Anne de Bellevue, Quebec: CEGEP John Abbott College, 1987.

- [Win86] Glynn Winskel. Event structures. In *Petri Nets: Central Models and Their Properties, Advances in Petri Nets 1986, Part II, Proceedings of an Advanced Course, Bad Honnef, Germany, 8-19 September 1986*, pages 325–392, 1986.