Modélisation à l'aide des systèmes de transition

Cours de langages de spécification

Catalin Dima

Configurations et symboles

- Les automates fonctionnent sur des alphabets contenant des symboles atomiques.
- Comment arrive-t-on à modéliser le comportement des systèmes informatiques avec de tels symboles ?
- Symbole = état des variables :
  - Supposons que V est l'ensemble des variables du programme.
  - Supposons que chaque variable prend des valeurs *booléennes*.
  - Une configuration du système est une valuation des variables : $V \longrightarrow \{0, 1\}$.
  - L'alphabet devient alors l'ensemble des valuations des variables.
- Des variables à valeurs entières bornées (dans un intervalle $[0..k]$) peuvent être *codés* dans les variables à valeurs booléennes.
- Par contre ce modèle ne permet pas de coder des variables à valeurs non-bornées !

Systèmes de transition (Kripke)

Système de transition (ou structure de Kripke) : tuple $T = (Q, V, Q_0, \delta, \pi)$ où :

- Q est un ensemble fini d'états.
- V est un ensemble de variables booléennes.
- $\delta \subseteq Q \times Q$ est la relation de transition.
- $\pi : Q \longrightarrow 2^V$ est l'étiquetage des états par des variables booléennes.
  - Si $v \in \pi(q)$, on dit que v est vraie dans q.
  - Autre variante de définition : $\pi \subseteq Q \times V$, indiquant, si $(q, v) \in \pi$, que v est vraie dans q.
  - Assez souvent, π remplace carrément Q, les états étant caractérisés par les valuations des variables.

Exemple ...

Comparaison avec les automates de Moore

- Ensemble d'états dans un système de transition = ensemble d'états dans un automate.
- Fonction d'étiquetage des états = alphabet.
- Relation de transition identique.
- Pas d'états finaux (on verra pourquoi).
- Remarquer qu'un système de Kripke dans lequel les états sont caractérisés par π correspond à un automate de Moore déterministe !

Le loup, la chèvre et le chou

- Un chasseur amenant un loup, une chèvre (vivants) et un chou, arrive aux berges d'une rivière, et veut passer avec les deux animaux et le chou de l'autre coté.
- Un ferry est amarré pour permettre la traversée, mais ne possède que deux places.
- Si le chasseur laisse le loup avec la chèvre sur une berge, le loup mange la chèvre.
- Pareil, si la chèvre reste sur une berge avec le chou et n'est pas surveillée par le chasseur, elle mange le chou.
- Le chasseur veut que tout son bagage vivant ou pas passe de l'autre coté de la rivière en bon état ou bonne santé.

Système de Kripke pour le loup, la chèvre et le chou

- Variables $V = \{Chass, Loup, Chev, Chou\}$.
- Interprétations : $var = 0$ ssi personne/objet sur rive droite, sinon 1.
  - États : quadruplets de valeurs $(Chass, Loup, Chev, Chou)$.
- Transitions :
  - Passage du chasseur et du loup d'une berge à l'autre : $(i_1, i_2, i_3, i_4) \longrightarrow (1 - i_1, 1 - i_2, i_3, i_4)$.
  - Passage du chasseur et de la chèvre d'une berge à l'autre : $(i_1, i_2, i_3, i_4) \longrightarrow (1 - i_1, i_2, 1 - i_3, i_4)$.
  - Etc.
  - Traversée "instantanée" (abstraction) $\longrightarrow$ pas de valeur spéciale pour les variables "pendant" la traversée.
- État initial ?
- États interdits ?
- Solution ?

Spécification du système de Kripke

Le loup, la chèvre et le chou

Formule définissant la relation de transition :

- (Valeurs des) variables avant la transition : $Chass$, $Loup$, $Chev$, $Chou$.
- (Valeurs des) variables après la transition : $Chass'$, $Loup'$, $Chev'$, $Chou'$.

Variantes :

1. Affectation des variables primes impliquant les variables non-primes :

$$\begin{aligned} Chass' = 1 - Chass \wedge (Loup' = 1 - Loup \wedge Chev' = Chev \wedge Chou' = Chou) \vee \\ (Loup' = Loup \wedge Chev' = 1 - Chev \wedge Chou' = Chou) \vee \\ (Loup' = Loup \wedge Chev' = Chev \wedge Chou' = 1 - Chou) \vee \\ (Loup' = Loup \wedge Chev' = Chev \wedge Chou' = Chou) \vee \end{aligned}$$

2. Autre variante :

$$Chass' = 1 - Chass \wedge Loup' + Chev' + Chou' \leq 1 \wedge Loup' \geq 0 \wedge Chev' \geq 0 \wedge Chou' \geq 0$$

Passage à niveau

- Capteur d'approche du train : variable $pres \in \{0, 1\}$.
- Distance entre le train et le passage à niveau : variable $d \in \{-3, -2, -1, 0, 1, 2, \infty\}$.
- Contrôleur de la barrière : accès à la variable *pres* et variable supplémentaire $barr \in \{0, 1\}$.
- Modélisation par deux composants : le train et (le contrôleur de) la barrière.
- Système complet = composition synchrone des deux composants !
  - Intersection des langages !

Train

- Le train approche le passage à niveau de loin :

$$\phi_1 : pres = 0 \wedge d = \infty \wedge pres' = 1 \wedge d = 3$$

- Ensuite il avance mètre par mètre :

$$\phi_2 : pres = pres' \wedge d' = d + 1 \wedge d \leq 2$$

- Lorsqu'il est à 2m après la barrière, une seconde après il est loin :

$$\phi_3 : pres = 1 \wedge pres' = 0 \wedge d = 2 \wedge d' = \infty$$

- Lorsqu'il est loin, il peut aussi rester loin :

$$\phi_4 : pres = pres' = 0 \wedge d = \infty$$

- Relation de transition de la composante "train" :

$$\phi_1 \wedge \phi_2 \wedge \phi_3 \wedge \phi_4$$

- Dessin...

Barrière

- Lorsque le train vient de loin, le capteur *pres* change de valeur :

$$\psi_1 : pres = 0 \wedge pres' = 1 \wedge barr' = barr = 0$$

- Une seconde après avoir capté l'approche du train, la barrière est baissée *barr* change de valeur :

$$\psi_2 : pres = pres' = 1 \wedge barr = 0 \wedge barr' = 1$$

- Lorsque le train s'éloigne, le capteur *pres* change à nouveau de valeur :

$$\psi_3 : pres = 1 \wedge pres' = 0 \wedge barr = barr' = 1$$

- Une seconde après avoir capté l'éloignement du train, la barrière est levée :

$$\psi_4 : pres = pres' = 0 \wedge barr = 1 \wedge barr' = 0$$

- Enfin, tant que le capteur ne signale aucun changement, la barrière reste dans sa position précédente :

$$\psi_5 : pres = pres' \wedge barr = barr'$$

- Relation de transition de la composante "barrière" :

$$\phi_1 \wedge \phi_2 \wedge \phi_3 \wedge \phi_4 \wedge \phi_5$$

- Dessin...

Composition des systèmes, cas des variables partagées

- Intersection des deux automates (en fait systèmes de Kripke).
- ... sauf que les alphabets des deux systèmes ne sont pas les mêmes !
  - Les variables de la composante “train” ne sont pas les mêmes que celles de la composante “barrière”.
- Ajoutons dans chaque composante la variable qui manque !
  - Chaque état est dupliqué (explosé) en deux autres, pour chaque valeur de la variable rajoutée.
- Adaptons les transitions !
  - Chaque transition $q \longrightarrow q'$ donne lieu à quatre transitions $(q, i) \longrightarrow (q', j)$ $(i, j \in \{0, 1\})$.
- Maintenant on peut construire l'intersection.
  - L'intersection peut se calculer directement aussi, sans éclatement intermédiaire.
- Dessin...

Problème d'exclusion mutuelle

- On a deux programmes qui possèdent chacun une section critique :
  - Lecture-écriture dans un fichier commun.
- Pour la cohérence des données dans le fichier, à tout moment, au maximum un programme peut se trouver dans sa section critique.
  - C'est à dire, tant qu'un programme n'a pas quitté sa section critique, l'autre ne peut pas encore rentrer dans la sienne.
  - Les programmes ne peuvent pas être exécutés de manière anarchique !
- On suppose que les programmes sont exécutés sur une machine uni-processeur.
  - Donc à chaque instant une seule instruction peut s'exécuter.

Algorithme de Peterson

Exclusion mutuelle sans interblocage

```
while (true) {
   flag1 := true;
   turn := 2;
   if !flag2 || (turn=1)
      section critique 1
   flag1 := false ;
}
```

```
while (true) {
   flag2 := true;
   turn := 1;
   if !flag1 || (turn=2)
      section critique 2
   flag2 := false ;
}
```

Propriétés désirés :

- Exclusion mutuelle : les deux programmes ne peuvent pas se trouver en même temps en section critique.
- Non-interblocage : à tout moment, un des deux programmes peut exécuter une de ses instruction.

Comment les prouver ?

Traduction des programmes à variables partagées en systèmes de Kripke

- Variables du système de Kripke = variable du programme.
- États = étiquettes des instructions du programme + valeurs des variables.
- Les transitions modélisent l'évolution du programme :
  - L'exécution d'une instruction fait avancer le système de Kripke de l'état correspondant à l'instruction courante à l'état correspondant à l'instruction suivante.
  - L'instruction suivante peut ne pas être l'instruction qui suit, dans le cas d'une instruction conditionnelle !
  - Les valeurs des variables sont mises à jour selon l'instruction exécutée.

Traduction des programmes à variables partagées en systèmes de Kripke

- Mais il y a des variables partagées !
  - Ex. *turn*, *flag1*, *flag2*.
  - Prog2 peut changer *turn*, *flag2* !
- Alors il faut prévoir aussi la possibilité de changer d'état par seule modification de valeur de variable partagée par l'autre programme.
  - Donc sans changer d'étiquette d'instruction de programme !
- Dessin (un pour chaque programme !)

Composition semi-synchrone

- Lorsqu'on a plusieurs programmes qui doivent s'exécuter sur un ordinateur uni-processeur, les états des programmes se succèdent en entrelacement.
- L'état du système est composé du produit cartésien des états (étiquettes) de chaque programme, des valuations des variables partagées et des valuations des variables propres.
- Ça ressemble beaucoup au shuffle, sauf que l'ensemble des variables de la composition est l'union des ensembles de variables de chaque programme.
- Cela implique que le système de Kripke correspondant à chaque programme soit "éclaté" !

Éclatement d'un système de Kripke

Étant donné un système $T = (Q, V, Q_0, \delta, \pi)$ et un ensemble de variables $V' \supseteq V$, on construit $\overline{T} = (\overline{Q}, \overline{V}, \overline{Q_0}, \overline{\delta}, \overline{\pi})$ où :

- Chaque état est éclaté en autant d'états que de valuations des variables supplémentaires :

$$\overline{Q} = Q \times 2^{V'}$$
$$\overline{Q_0} = Q_0 \times 2^{V'}, \qquad \overline{Q_f} = Q_f \times 2^{V'}$$
$$\overline{\delta} = \left\{(q, v) \longrightarrow (q', v') \mid q \longrightarrow q' \in \delta, v, v' \subseteq V'\right\}$$
$$\overline{\pi}(q, v) = \pi(q) \cup v$$

- Résultat : toute trace acceptée par $\overline{T}$ est un éclatement d'une trace acceptée par T.
  - C.à.d. chaque valuation des variables de V est augmentée d'une valuation (quelconque!) de variables dans V'.

Composition semi-synchrone de deux systèmes

- Étant données deux structures de Kripke $T_1 = (Q_1, V_1, Q_0^1, \delta_1, \pi_1)$ et $T_2 = (Q_2, V_2, Q_0^2, \delta_2, \pi_2)$, on construit $T = (Q, V, Q_0, \delta, \pi)$ comme étant :

$$T_1 \otimes T_2 = \overline{T}_1 \cap \overline{T}_2$$

  - C.à.d. l'intersection de l'éclatement des deux systèmes !
- L'algorithme de Peterson en tant que composition semi-synchrone de systèmes de Kripke :
  - Chaque programme sera modélisé en système de Kripke, puis éclaté.
  - Puis on calcule la composition semi-synchrone des deux éclatements.
  - Dessin (grand !)...

Quelques conclusions

Les bienfaits de la théorie des automates (ou leurs équivalents, les systèmes de Kripke) pour la modélisation des systèmes :

- État $\sim$ valeurs des variables à un instant donné.
- Modules = composition (synchrone ou semi-synchrone).
- Passage du temps de durée d : suite de d transitions dans le système.
- Certains problèmes de recherche de solution se résolvent à l'aide de modèles de systèmes de transitions.
- La correctitude de certaines solutions se vérifie à l'aide de modèles de systèmes de transitions.
- Le travail de vérification implique un travail de traduction (modélisation) en "langage" de type système de transition.
  - Vrais (et puissants) langages de modélisation ?
- Les propriétés vérifiées sont assez simples : atteignabilité, sûreté, absence d'interblocage.
  - Autres propriétés plus puissantes ?
  - Vrai langage de spécification des propriétés à vérifier ?