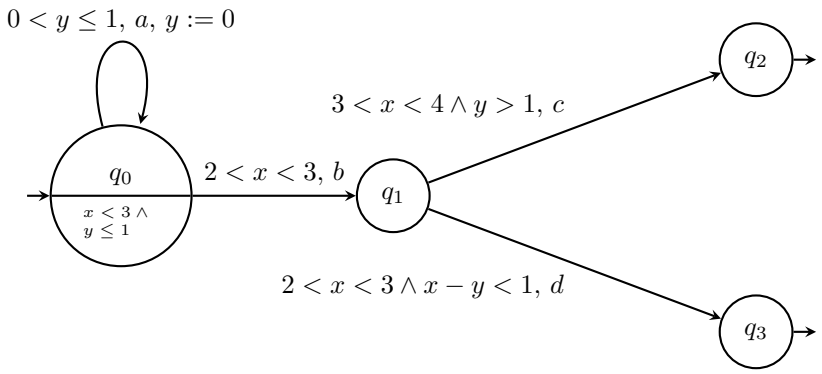
Automates temporisés

TD/TME 1 : Modélisation par automates temporisés et hybrides – Initiation à HyTech

Exercice 1 Sémantique des automates temporisés

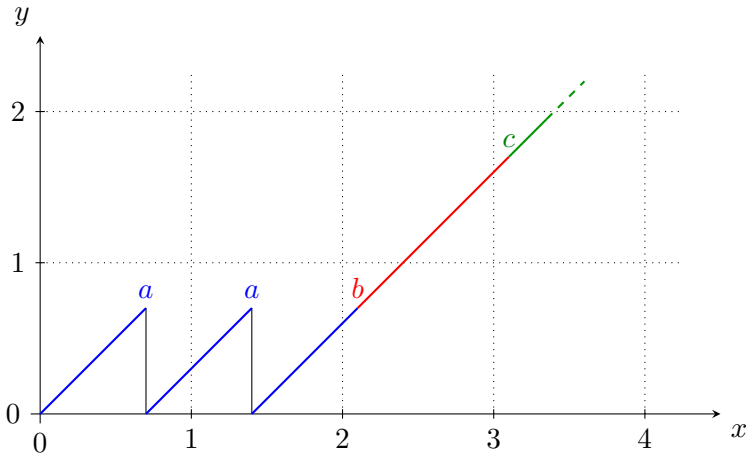
On considère l'automate temporisé suivant :



1. Donner une exécution de cet automate qui atteint l'état q_2.
2. Représenter dans le plan les valeurs successives des horloges durant cette exécution.

Solution de l'exercice 1

1. $\rho = (a, 0.7), (a, 1.4), (b, 2.1), (c, 3.1)$ ou encore $\sigma = (a, 0.5), (a, 1.5), (b, 2.5), (c, 3.1)$.
2. On va suivre les valeurs successives des horloges lors de l'exécution de ρ. On tracera le parcours en bleu lorsque l'on se trouve dans l'état q_0, en rouge dans q_1 et en vert dans q_2.

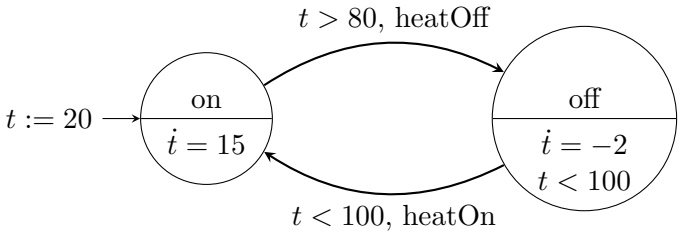


Exercice 2 Modélisation d'une machine à café

1. Modéliser par un automate temporisé la machine à café répondant aux spécifications suivantes :
   – Lorsqu'une pièce est insérée, si rien ne se passe au bout de 10 secondes, elle est rendue.
   – Une pièce peut être rendue si l'utilisateur appuie sur le bouton de remboursement.
   – Lorsqu'une pièce est présente dans la machine, l'utilisateur peut demander un café.

– Un café prend 30 secondes à se faire. Pendant ce temps, aucune action n'est possible de la part de l'utilisateur.

2. On veut affiner la modélisation de la machine à café en prenant en compte le montant présent dans la machine. Pour cela, on va utiliser une extension des automates temporisés, appelés *automates hybrides*. Dans ces automates, les horloges sont des variables pouvant varier à des vitesses différentes les unes des autres, et différemment selon les états. Des mises à jour peuvent aussi donner une nouvelle valeur (constante) à une variable, et non seulement de remises à zéro.
Par exemple, l'automate hybride suivant représente le chauffe-eau de la machine à café. Il est soit allumé soit éteint. La température de l'eau est modélisée par la variable t. Elle croît de 15°C par minute lorsque le chauffe-eau est allumé ($\dot{t} = 15$), et décroît de 2°C par minute lorsque le chauffe-eau est éteint ($\dot{t} = -2$). Initialement, l'eau est à 20°C, et elle doit être maintenue entre 80 et 100°C.

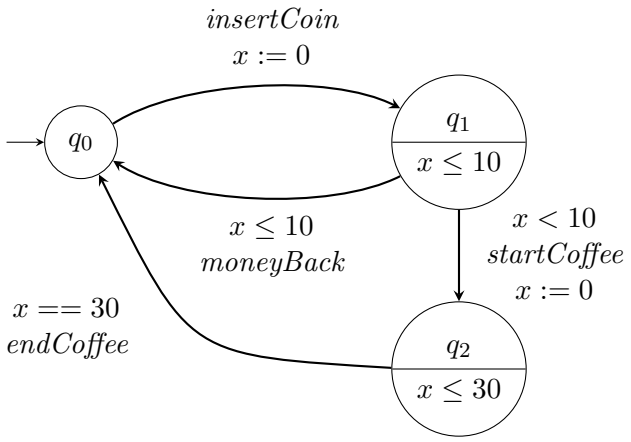


Modéliser la machine à café en utilisant une variable qui garde la valeur du montant présent dans la machine, initialement 0 centimes. On supposera qu'il existe de pièces de 5, 10, 20, et 50 centimes et donc des actions associées *insert5*, *insert10*, etc. Le café coûte 40 centimes. On pourra supposer que changer la valeur de la somme présente dans la machine prend une seconde.

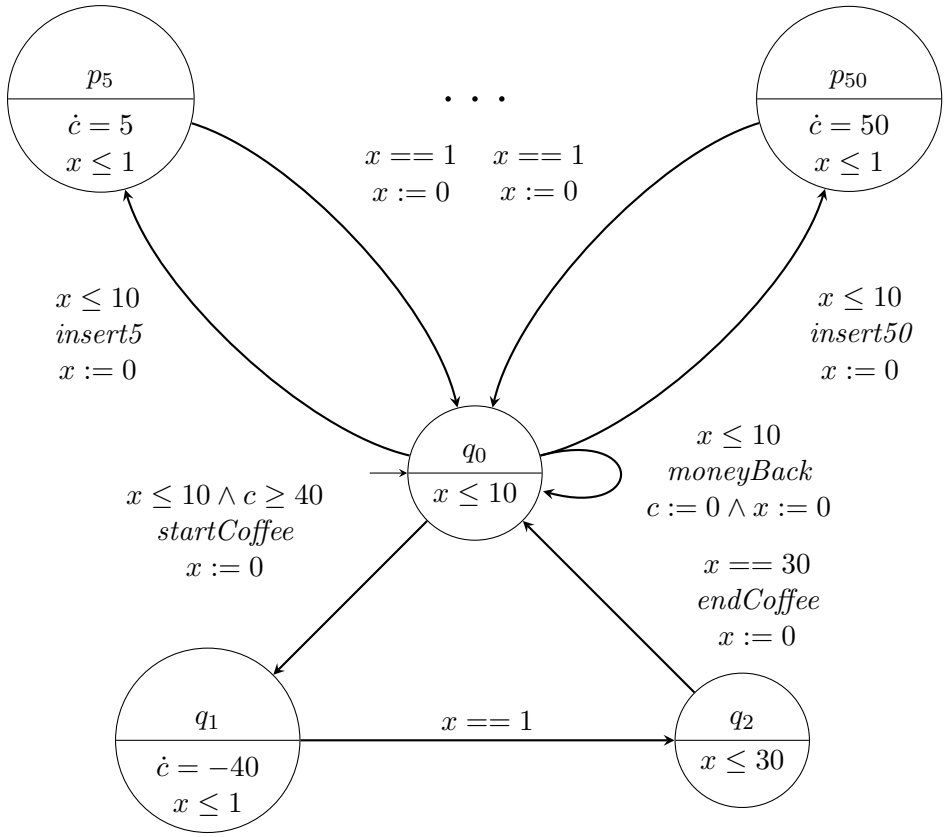
3. Modifier l'automate hybride obtenu à la question précédente pour qu'il prenne en compte la quantité de café en grain et moulu qu'il contient. On supposera qu'il faut 50g de café moulu pour faire un café et que le moulin produit 7g de café à la seconde ; il y a initialement 1kg de café en grain et pas de café moulu dans la machine. De plus, on préfèrera toujours que le café soit moulu au dernier moment. On pourra commencer par modéliser le moulin à café séparément.

Solution de l'exercice 2

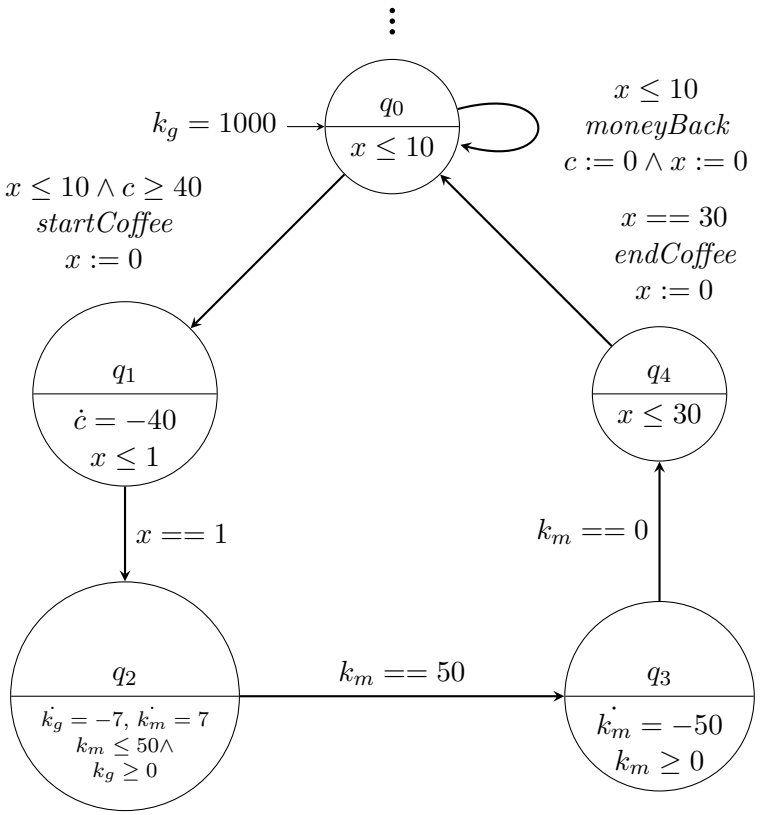
1. Les actions utilisées seront
– *insertCoin* quand un pièce est insérée,
– *startCoffee* quand le café commence à se préparer,
– *endCoffee* quand le café est prêt,
– *moneyBack* quand le client est remboursé.
On aurait aussi pu séparer les cas où le client demande le remboursement et où il lui est imposé par *timeout*.



2. La variable c représentera le montant présent dans la machine. Quand ce n'est pas précisé, $\dot{x} = 1$ et $\dot{c} = 0$.



3. Les variables k_g et k_m représenteront respectivement la quantité de café en grain et moulu présent dans la machine. Quand ce n'est pas précisé, $\dot{x} = 1$ et $\dot{c} = \dot{k_g} = \dot{k_m} = 0$.



Utilisation de HyTech

Le logiciel HyTech permet de vérifier des propriétés, en particulier d'accessibilité sur les automates hybrides. Il a été développé dans les années 1990 par l'équipe de Henzinger (l'inventeur des automates hybrides) et n'a que peu évolué depuis.

Installation. L'archive de HyTech modifiée pour les machines de l'ARI est disponible à l'adresse suivante : `http://lip6.fr/Mathieu.Sassolas/enseignement/hytech.tar.gz`. Pour l'installer, on décompresse l'archive, compile le programme, et crée un alias afin de pouvoir utiliser HyTech plus simplement :

```
tar -xvzf hytech.tar.gz
cd HyTech/src/
make
alias hytech='~/HyTech/src/hytech.exe'
```

HyTech ne dispose malheureusement que d'une interface textuelle. On l'appellera donc directement depuis un terminal, sur un fichier contenant les déclaration d'automates : `hytech mon_fichier.hy`. Les options `-p0`, `-p1` et `-p2` spécifient le niveau de verbosité. Le fichier `.hy` sera édité par votre éditeur de texte préféré. Il est conseillé d'utiliser un fichier par système, HyTech composant automatiquement tous les automates du fichier [1].

1. Plus d'explication à ce sujet au Cours/TD/TME suivant

Syntaxe. Pour mettre des commentaires, on utilise deux tirets : "`--`". La fin de la ligne est alors ignorée.

La définition d'un système commence par les variables, et horloges du système :

```
var
    x,            -- Temps depuis la dernière extinction
    y: clock;     -- Temps total
    t: analog;    -- Température
    z: stopwatch; -- Temps passé allumé
```

Les horloges (*clock*) évoluent toutes avec pente 1 dans tout l'automate. Les variables physiques (*analog*) non évoluent avec une pente variable selon les états ; elle devra être spécifiée dans chaque état. Les chronomètres (*stopwatch*) sont des horloges que l'on peut arrêter (ou de manière équivalente des variables ayant pour pente 1 ou 0). On peut définir plusieurs variables du même type en les séparant par une virgule.

On définit ensuite chaque automate séparément en spécifiant un nom, des étiquettes de synchronisation, et une configuration initiale :

```
automaton chauffe_eau
    synclabs: start,
              stop;
    initially allume & t=20;

  ⋮

end
```

Ici, on spécifie les étiquettes *start* et *stop* ; on précise que l'état initial est *allume* et que la température est de 20°C. La définition de l'automate se terminera par `end`.

Pour définir chaque état de l'automate, on spécifie ses invariants, la vitesse des variables, et ses transitions sortantes :

```
loc allume: while t<100 wait {dt=15,dz=1}
    when t>80 do {x'=0} sync stop goto eteint;
```

L'état contient un invariant (`t<100`) et la pente des variables physiques (`dt=15`, et `dz=1`) ; avec les notations précédentes cela correspond à $\dot{t} = 15$ et $\dot{z} = 1$. Les différentes transitions contiennent garde (`t>80`), ou affectation (`x'=0`, qui correspond à $x := 0$), étiquette (`stop`) et état cible (`eteint`). Les gardes et les invariants peuvent être des conjonctions de comparaisons à l'aide du symbole `&`. On séparera les contraintes d'encadrement d'une horloge en une conjonction de deux contraintes : $0 \leq x < 1 \equiv x \geq 0 \wedge x < 1$. Une absence de condition sera spécifiée par `True`. Plusieurs affectations peuvent avoir lieu, séparées par des virgules.

Une fois les automates définis, on peut demander à HyTech de faire des vérifications d'accessibilité de régions.

```
var
    init,
    facture,
    accessible: region;
```

```
init:= loc[chauffe_eau]=allume & t=20 & z=0 & x=0 & y=0;
facture:= z > 500;
accessible:= reach forward from init endreach;

if empty(accessible & facture)
    then prints "On reste toujours allumé moins de 10 minutes";
    else prints "On peut rester allumé plus de 10 minutes";
         print trace to facture using accessible;
endif;
```

On définit ici une région initiale, où le chauffe eau est allumé et à 20°C, une région que l'on cherche à atteindre, où le chauffe eau a été allumé plus de 10 minutes, et enfin l'ensemble des régions accessibles depuis la région initiale. On fait ensuite le test du vide de l'intersection de *facture* de *accessible*, afin de savoir si la région cible est accessible. Dans ce cas on demande à afficher une exécution menant vers la région *facture*. La commande `prints` affiche des chaînes de caractères tandis que `print` affiche l'objet (région, trace,...) passé en argument.

Attention ! Sur les automates hybrides, l'accessibilité ne termine pas toujours. De plus, l'accessibilité peut se calculer en avant (*forward*) ou en arrière (*backward*). Ces deux méthodes peuvent terminer ou non indépendemment l'une de l'autre.

Exercice 3 Prise en main de HyTech

1. Transcrire l'automate de l'exercice 1 en HyTech et vérifier que l'on peut effectivement atteindre q_2.
2. Transcrire le dernier automate de l'exercice 2 en HyTech. Afin de garder un nombre fini de configurations, on limitera la somme présente dans la machine (par exemple à 10€). Vérifier que l'on peut atteindre une configuration où il n'y a plus de café.

Exercice 4 Évacuation de l'eau dans une mine

Dans une mine se trouve une pompe qui évacue de l'eau. Quand la pompe est arrêtée, le niveau de l'eau augmente de 90L à l'heure. Quand elle fonctionne, le niveau de l'eau baisse de 30L à l'heure. On considère seuil bas de 10L où l'on doit s'arrêter de pomper dans les 5 minutes afin de ne pas abîmer la pompe, et un seuil haut de 1000L où l'on doit absolument démarrer la pompe dans les 5 minutes, sans quoi la mine serait innondée. Paralèlement au problème de l'eau, dans cette mine il y a parfois du gaz, qui pourrait exploser si la pompe est en fonctionnement lorsque du gaz est présent. Lorsque du gaz est détecté, la pompe doit s'arrêter dans la minute. Le gaz s'évacue tout seul en au plus 5 minutes, et ne revient au plus que toutes les 10 minutes.

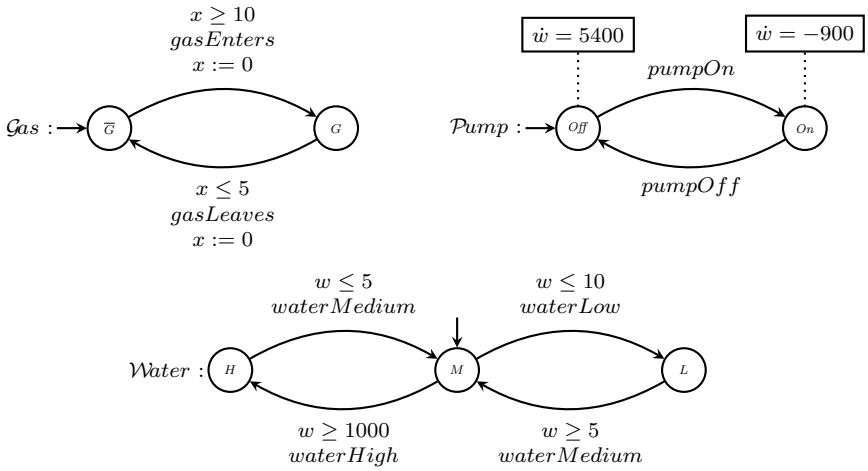
On construira tous les automates dans le même fichier et on prendra soin de donner le même nom à deux action identiques dans deux automates différents.

1. Modéliser un système représentant le niveau de l'eau, un autre représentant le présence ou l'absence de gaz, et un dernier pour la pompe.
2. Modéliser un contrôleur qui allume ou éteint la pompe en tenant compte de la quantité de l'eau et de la présence de gaz. On préfèrera innonder la mine que de la faire exploser.

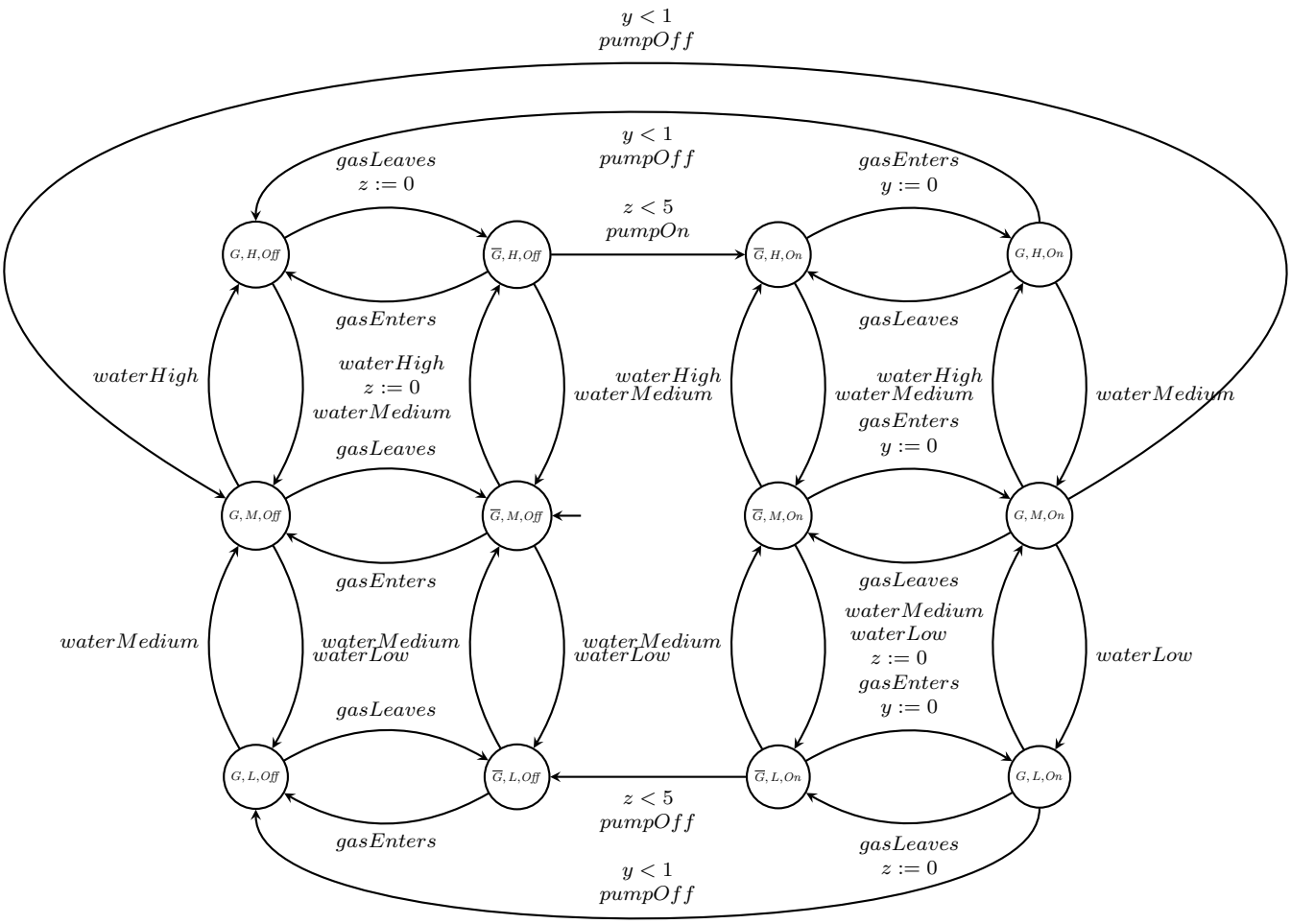
3. En laissant HyTech composer les différents automates, vérifier si la mine peut être innondée.

Solution de l'exercice 4

1. On utilise une variable hybride w pour garder en permanence la valeur du niveau d'eau dans la mine. Des signaux sont envoyés lorsqu'il y a un changement de l'état de la mine : présence ou absence de gaz, changement de niveau d'eau, allumage ou extinction de la pompe...



2. L'idée générale derrière le contrôleur est qu'il recevra des messages venant de $\mathcal{G}as$ et $\mathcal{W}ater$ et enverra des messages à $\mathcal{P}ump$. Il n'a pas besoin de reprendre les gardes déjà implémentées par les sous systèmes de la question précédente. Le contrôleur appliquera la stratégie suivante :
   – dès que l'on a du gaz on éteint la pompe
   – on allume la pompe quand on est au niveau haut, et on ne l'éteint que quand on est au niveau bas
   – si il y a du gaz et un niveau d'eau haut, on n'allume pas la pompe pour autant, et on l'éteint si elle est déjà allumée


$y < 1$
$pumpOff$
$y < 1$
$pumpOff$
$gasLeaves$
$z := 0$
$gasEnters$
$y := 0$
$z < 5$
$pumpOn$
G, H, Off
$\overline{G}, H, Off$
$\overline{G}, H, On$
G, H, On
$gasEnters$
$gasLeaves$
$waterHigh$
$waterHigh$
$z := 0$
$waterMedium$
$waterHigh$
$waterMedium$
$waterHigh$
$waterMedium$
$gasEnters$
$y := 0$
$waterMedium$
$gasLeaves$
G, M, Off
$\overline{G}, M, Off$
$\overline{G}, M, On$
G, M, On
$gasEnters$
$gasLeaves$
$waterMedium$
$waterMedium$
$waterLow$
$waterMedium$
$waterLow$
$waterMedium$
$waterLow$
$z := 0$
$gasEnters$
$y := 0$
$waterLow$
$gasLeaves$
G, L, Off
$\overline{G}, L, Off$
$\overline{G}, L, On$
G, L, On
$z < 5$
$pumpOff$
$gasEnters$
$gasLeaves$
$z := 0$
$y < 1$
$pumpOff$