Transformation de modèle à modèle

Attention : Version d'Eclipse à utiliser : `/mnt/n7fs/ens/tp_cregut/eclipse-gls/eclipse`

Exercice 1 : Transformation M2M, SimplePDL vers PetriNet

1.1. Indiquer comment transformer un processus (élément `Process`), puis une activité (élément `WorkDefinition`) et, enfin, une dépendance (élément `WorkSequence`).

1.2. Dessiner le réseau de Petri qui correspond au modèle de processus de la figure 1 qui définit deux activités A1 et A2 reliées par une relation de précédence de type `finishToStart`. Ainsi, A1 doit être terminée avant que A2 puisse commencer.

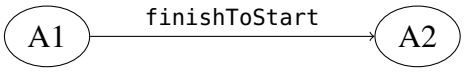

FIGURE 1 – Exemple de processus

Indiquer comment il faut modifier le réseau de Petri si on transforme la relation `finishToStart` en une relation `startTostart` (l'activité A2 ne peut être commencée que si l'activité A1 est commencée) ou `startToFinish` (l'activité A2 ne peut se terminer que l'activité A1 est commencée).

1.3. *Créer un nouveau projet ATL.* Dans votre *Workspace* eclipse, faire un clique droit puis *New / Other... / ATL / ATL Project*. Lui donner un nom (par exemple : `fr.n7.simplepdl2petrinet`).

1.4. Récupérer et importer dans le projet ATL l'amorce du module ATL (listing 1). Il contient une règle pour transformer un élément de type `Process` d'un modèle SimplePDL en un élément de type PetriNet. La règle qui transforme une `WorkDefinition` a juste été commencée.

1.5. Adapter la transformation ATL fournie à votre métamodèle des réseaux de Petri.

1.6. *Exécuter le module ATL.* Pour exécuter une transformation ATL, il faut définir sa configuration de lancement. Pour cela, aller dans le menu *Run / Run Configurations...*. Dans la colonne de gauche trouver la section *ATL Transformations*, la sélectionner puis à l'aide d'un clique droit, cliquer sur *New*. Donner un nom à la configuration ainsi créée. Il vous faut ensuite remplir le champ *ATL Module* afin de sélectionner le module ATL à lancer. Une fois cela fait, les autres champs de la configuration vont s'activer, il vous faut alors les renseigner (renseigner les métamodèles et les modèles d'entrée et de sortie avec le bouton *WorkSpace...*). Ensuite, dans l'onglet *Advanced*, cocher *Clear console before launch* et *Print execution times to console*. Finaliser la création de la configuration en cliquant sur *Apply* puis *Run* pour lancer la transformation.

1.7. Compléter le module qui transforme un modèle de processus conforme à SimplePDL en un modèle de réseau de Petri conforme à PetriNet. On construira progressivement ce module en le testant au fur et à mesure. Ainsi, on commencera par transformer les éléments `Process` (déjà fait), puis on traitera les éléments de type `WorkDefinition` et enfin les éléments de type `WorkSequence`.

Listing 1 – Le module SimplePDL2PetriNet

```
1  -- @nsURI SimplePDL=http://simplepdl
2  -- @nsURI PetriNet=http://petrinet
3
4  module SimplePDL2PetriNet;
5  create OUT: PetriNet from IN: SimplePDL;
6
7  -- Obtenir le processus qui contient ce process element.
8  -- Remarque: Ce helper ne serait pas utile si une référence opposite
9  -- avait été placée entre Process et ProcessElement
10 helper context SimplePDL!ProcessElement
11 def: getProcess(): SimplePDL!Process =
12    SimplePDL!Process.allInstances()
13       ->select(p | p.elements->includes(self))
14       ->asSequence()->first();
15
16 -- Traduire un Process en un PetriNet de même nom
17 rule Process2PetriNet {
18    from p: SimplePDL!Process
19    to pn: PetriNet!PetriNet (name <- p.name)
20 }
21
22 -- Traduire une WorkDefinition en un motif sur le réseau de Petri
23 rule WorkDefinition2PetriNet {
24    from wd: SimplePDL!WorkDefinition
25    to
26       -- PLACES d'une WorkDefinition
27       p_ready: PetriNet!Place(
28             name <- wd.name + '_ready',
29             marking <- 1,
30             net <- wd.getProcess()), -- une virgule entre les éléments créés
31       p_started: PetriNet!Place(
32             name <- wd.name + '_started',
33             marking <- 0,
34             net <- wd.getProcess())
35 }
```