Chaîne de vérification de modèles de processus
Rapport BE GLS

Robin Xambili
Youssef Bendagha

9 novembre 2017

Table des matières

Chapitre 1

Présentation du sujet

1.1 Introduction

Ce BE consiste à produire une chaîne de vérification de modèles de processus SimplePDL dans le but de vérifier leur cohérence, en particulier pour savoir si le processus décrit peut se terminer ou non. Pour répondre à cette question, nous utilisons les outils de model-checking définis sur les réseaux de Petri au travers de la boîte à outils Tina. Il nous faudra donc traduire un modèle de processus en un réseau de Petri.
Les TP présenteront les outils qui devront être utilisés pour réaliser ce BE. Chaque TP traite un aspect développé dans le BE. Le TP est une introduction qui devra être complétée en consultant au moins la documentation des outils utilisés. Le BE correspond à ce qui est fait dans les TP avec une petite extension (les ressources).
De ce fait, ce rapport consistera en grande partie en un compte-rendu des différents TPs.

1.2 Objectifs du BE

Le but de ce BE consiste pour l'essentiel à définir la chaîne de vérification de modèles de processus sur une version simplifiée des modèles de processus. Ce travail est fait en TP, les principales étapes étant les suivantes :
1. Définition des métamodèles avec Ecore.
2. Définition de la sémantique statique avec OCL.
3. Utilisation de l'infrastructure fournie par EMF pour manipuler les modèles.
4. Définition de transformations modèle à texte (M2T) avec Acceleo, par exemple pour engendrer la syntaxe attendue par Tina à partir d'un modèle de réseau de Petri ou engendrer les propriétés LTL à partir d'un modèle de processus.
5. Définition de syntaxes concrètes textuelles avec Xtext.

6. Définition de syntaxes concrètes graphiques avec Sirius.
7. Définition d'une transformation de modèle à modèle (M2M) avec Java et avec ATL.
Ce travail fait en TP sera complété par la validation de la chaîne de transformation et l'ajout de ressources au langage de description de processus.

Chapitre 2

Compte-rendu des différents TPs

2.1 TP1 : La boite à outils Tina

La boîte à outils Tina propose plusieurs outils dont NetDraw et le model-checker selt.

2.1.1 L'outil NetDraw (nd)

L'outil nd (Net Draw) permet de visualiser un réseau de Petri et de le simuler. Nous avons pris comme exemple saisons.net ; grace à cet exemple nous avons apris à manipuler l'outil graphique nd.

2.1.2 Analyse par model-checking avec selt

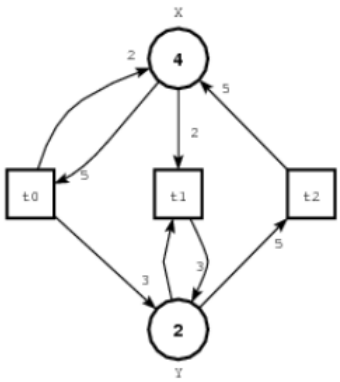
Les propriétés que l'on souhaite vérifier sur un réseau de Petri peuvent être exprimées en LTL (Logique Temporelle Linéaire), exemple du fichier saisons.ltl :
() <> Ete ; Toujours il y aura un été
- <> Ete ; Il n'y aura pas d'été
Ces proprietés peuvent etre verifiées en utlisant certaines lignes de commandes.

2.1.3 Détection des interblocages

Définition : Réseau de Petri bloqué => Un réseau de Petri est bloqué si aucune transition ne peut être tirée dans le marquage actuel. Un réseau est blocable s'il possède un marquage accessible bloqué, c'est-à-dire ne possédant pas de transition de sortie. Il existe différents critères de vivacité d'un réseau de Petri qui est alors qualifié de vivant (Live Petri Net).
L'outil Tina permet aussi de détecter les interblocages.

2.1.4 Création de réseau de Pétri

En appliquant nd sur la forme textuelle de Figure 1 (TP1) on obtient le graphe suivant :



2.1.5 Allocations de Ressource avec un client et deux gestionnaires de ressource

Un client (C_1) a besoin pour travailler de posséder de manière mutuellement exclusive deux ressources (R_1,R_2) qui sont distantes et gérées individuellement par deux gestionnaires (G_1,G_2).
La communication entre client et gestionnaire est régie par les messages suivants :

Req - demande d'accès à la ressource,

Ack - autorisation d'accès à la ressource,

Lib - libération de la ressource

Client C_1 : C1 demande séquentiellement les deux ressources. Plus précisément, il demande d'abord la ressource R_1, et lorsqu'il l'a obtenue, il demande la ressource R_2. Il entre en section critique lorsqu'il possède les deux ressources. Il libère ensuite en même temps les deux ressources et retourne à son état initial.

Gestionnaire G_j : À la réception d'un message Req envoyé par le client C_i, G_j alloue exclusivement la ressource à C_i (i.e. envoi du message Ack à C_i si la ressource est libre, sinon la requête reste en attente. À la réception du message Lib, la ressource est libérée.

2.1.6 Allocations de Ressource avec deux clients et deux gestionnaires de ressource

On ajoute au système précédent un second client (C_2) dont le comportement est décrit ci-dessous :

Client C_2 : C_2 demande séquentiellement les deux ressources. Plus précisément, il demande d'abord la ressource R_2, et lorsqu'il l'a obtenue, il demande la ressource R_1. Il entre en section critique lorsqu'il possède les deux ressources. Il libère ensuite en même temps les deux ressources et retourne à son état initial.

2.2 TP2 : Méta-modélisation et sémantique statique

2.2.1 Comprendre les outils de métamodélisation d'Eclipse EMF

L'objectif de ces premiers exercices est de prendre en main les outils proposés par Eclipse Modeling Project pour la métamodélisation. En fait, nous allons surtout utiliser les outils développés dans le cadre d'Eclipse EMF (Eclipse Modeling Framework) et l'éditeur graphique [1] Ecore.

Nous nous appuyons sur un langage simplifié de modélisation de processus de développement, appelé SimplePDL. Son métamodèle Ecore est présenté à la figure 1.

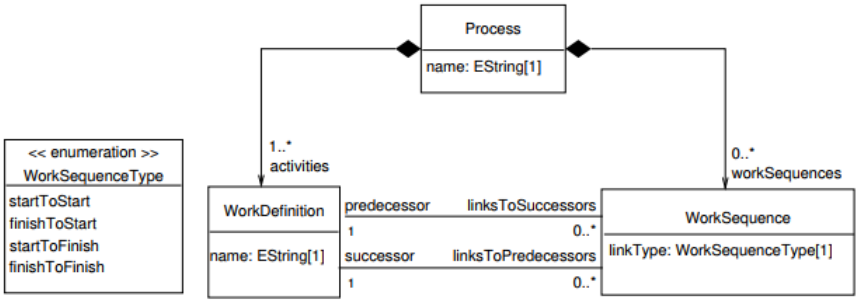


FIGURE 1 – Métamodèle initial de SimplePDL

Préparation A ce niveau là, nous nous sommes contenté de créer un projet et y placer le modèle fourni.

Visualisation du métamodèle Le métamodèle de SimplePDL est fourni sous la forme d'un fichier Ecore. Le fichier Ecore n'étant pas tres lisible, nous nous sommes tournés veers l'éditeur graphique afin de mieux visualliser notre métamodèle.