14/01/2018

# Rapport TP Uppaal: Exercice Gossiping Girls

Benjamin DUTRIAUX

# I – Introduction

Cet exercice a été réalisé dans le cadre d'un TP sur le logiciel Uppaal, développé conjointement par les universités d'Aalborg au Danemark et d'Uppsala en Suède, il permet de modéliser, simuler et vérifier des automates temporisés.

Le problème qui nous a été posé ici est celui des Gossiping Girls. Il consiste en un nombre de personne possédant chacune un secret et visant à le partager avec toutes les autres. Le but étant de voir en combien d'appels toutes les personnes peuvent connaitre tous les secrets, sachant qu'une personne ne peut en appeler qu'une seule autre à la fois et qu'à la fin de la conversation, les deux connaissent les secrets de l'autre.

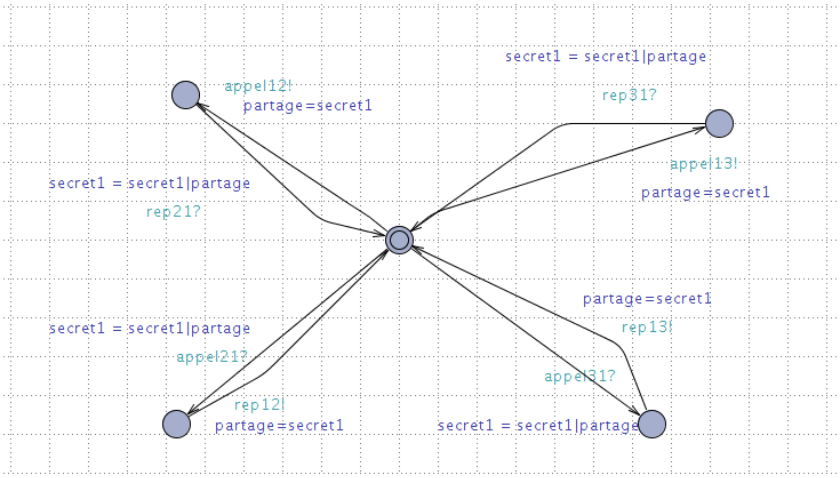
Cet exercice comporte 3 parties qui seront détaillées dans le chapitre II – Réalisation du TP.

# II – Réalisation du TP

## 1. *Modélisation avec un template par personne*

Cette modélisation du problème est la plus simple à réaliser, elle consiste à créer un template différent par personne et à les faire communiquer grâce à des canaux, un par couple possible pour les appels et pour les réponses aux appels. Dans cette modélisation, un secret est un entier qui est une puissance de deux, on crée aussi une variable globale « partage » qui sert de tampon entre les différents secrets. A chaque appel, la personne qui appelle met ses secrets connus dans la variable « partage », et celle qui est appelé effectue un « ou logique » entre la variable « partage » et ses secrets, ce qui fait qu'elle connait maintenant les secrets de l'autres sans avoir oublié les siens. Après l'appel, on fait l'inverse, c'est l'appelant qui met ses secrets dans la variable « partage » l'appelé qui effectue un « ou logique » entre cette variable et ses secrets. Ainsi, après un appel, les deux personnes connaissent les secrets de l'autre.

**1 : Automate de la personne 1**



Pour l'exercice, il aura fallu créer 3 template, modélisant 3 personnes. On lance alors le vérificateur avec la requête « E<>(Process1.secret1==7)&&( Process2.secret2==7)&&( Process3.secret3==7) », ce qui demande au vérificateur s'il est possible que les variables secret puissent toutes atteindre le chiffre 7, le chiffre 7 étant le résultat d'un « ou logique » entre toutes les variables secret initiales. Le simulateur répond alors que « la propriété est satisfaite », validant ainsi le modèle. On peut aussi lui demander le chemin le plus court pour accéder à ce résultat, ici il suffit de trois appels, dont un qui n'a pas besoin de réponse (donc 5 transitions) afin que toutes les personnes soient au courant de tous les secrets.

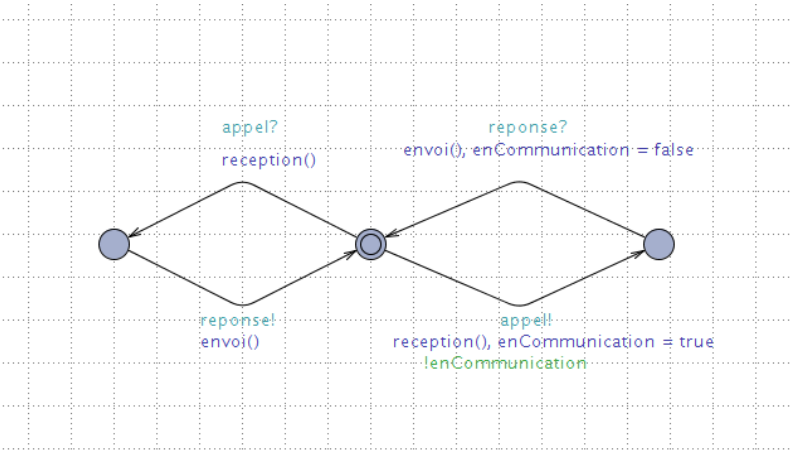
Cette modélisation est simple à mettre en place avec un nombre réduis de personnes, mais devient fastidieuse si le nombre de personne devient trop important, il faut recréer un template, des canaux à chaque fois et augmenter le nombre d'états de chaque template. On va donc voir dans la partie deux une solution qui permet d'éviter d'avoir à faire cela.

## 2. Modélisation avec un template unique

Afin de modéliser ce problème en pouvant changer simplement le nombre de personne, on va créer un template unique qui pourra communiquer avec une autre instance de ce même template. Chaque instance de ce template aura un variable « secret » qui sera une puissance de deux, pour cela, on utilisera le décalage binaire à gauche, on décale la variable à gauche le même nombre de fois que le numéro de la personne moins un afin de créer des puissances de deux à partir de $2^0$. On crée un canal « appel » et un canal « reponse », il faut aussi pouvoir passer un paramètre au template qui indiquera le numéro de la personne à laquelle il correspond. Le partage de secret fonctionne de la même façon que dans la première modélisation, on aura juste créé des fonctions afin de factoriser le code.

Je pensais que le modèle était bon car en effectuant la vérification avec 3 personnes, cela fonctionnait normalement, mais dès que l'on tentait de passer la vérification avec 4 personnes ou plus, le programme retournait une erreur. Ce qu'il se passait est qu'à partir de 4 personnes, il pouvait arriver qu'une personne en communication soit appelée par une autre personne. Pour pallier à cela, il fallait ajouter un booléen « enCommunication » qui passait à vrai lors d'un appel et qui repassait à faux à la fin de celui-ci, ainsi qu'une garde sur l'appel qui empêchait une personne d'appeler au cas où une communication était déjà en cours.

2 : Automate unique




Cette solution permet donc de modéliser le problème avec un nombre de personne théoriquement infini, mais elle ne prend pas en compte le temps que ces personnes mettent à effectuer l'appel. C'est ce que nous allons voit dans la troisième et dernière partie.

*Réponse à la question 3 :*

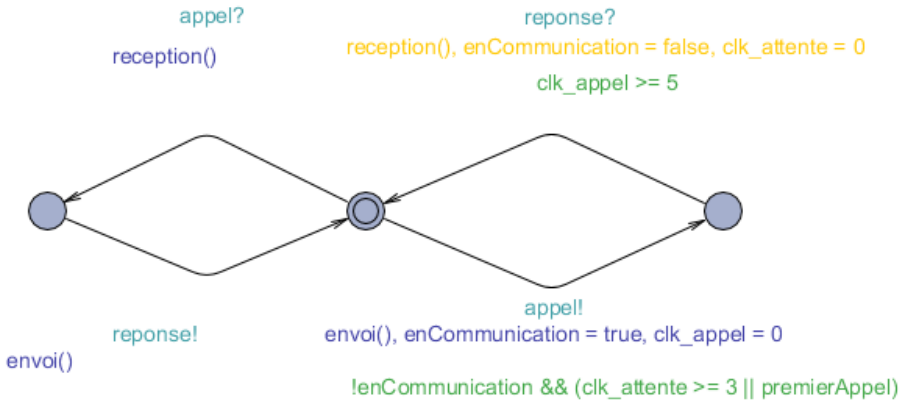
*Pour 6 personnes, il faut au minimum 8 appels, dont un qui n'a pas besoin de réponse, il y a donc besoin de 15 échanges.*

## 3. Modélisation incluant la temporalisation

Pour cette dernière partie, nous allons inclure la dimension temporelle dans la modélisation, pour cela, on a supposé un appel prenait 5 secondes et qu'une personne doit attendre 3 secondes entre deux appels effectué par lui-même.

Il faut donc ajouter des horloges, une qui comptabilise le temps lors d'une communication et qui se réinitialise au début d'un appel et l'autre qui vérifie que le temps d'attente entre chaque appel est bien de trois secondes et qui elle se réinitialise à la fin d'un appel. Il faut aussi vérifier que le premier appel n'ai pas besoin d'attendre trois secondes avant d'être lancé, il faut donc un booléen qui passe à faux une fois un appel effectué. On peut aussi rajouter une horloge globale pour voir le temps écoulé sans avoir à additionner la somme de chaque horloge.

3 : Automate incluant la temporalisation




Cette modélisation permet donc de connaitre le temps et le nombre d'appels nécessaires afin que tout le monde connaissent tous les secrets.

*Réponse à la question 4 :*

*Pour 4 personnes, il faut 20 secondes afin que tout le monde connaisse tous les secrets. 4 * 5 secondes d'appel et 0 seconde d'attente, le temps d'attente n'influe pas car il est plus court que le temps d'appel, il s'écoule donc lors des appels des autres personnes. Il y besoin ici de 4 appels, soit 8 échanges de secrets.*

## III – Difficultés

J'ai rencontré au cours de cet exercice certaines difficultés :

- L'utilisation des puissances de deux afin de simuler l'échange de secret, ce n'était pas très instinctif pour moi et j'ai mis un peu de temps à en comprendre le fonctionnement.
- Certaines erreurs ou j'ai inversé les échanges de secrets entre l'appel et la réception, qui sont des erreurs d'étourderie mais qui peuvent être assez longues à corriger.
- Le problème d'appels simultanés de l'exercice 2 qui a été résolu par l'ajout d'un booléen.

## IV – Conclusion

Cet exercice sur les gossiping girls a permis de synthétiser tous les cas vus dans les précédents exercices et hormis quelques difficultés citées plus haut, il n'aura pas été si compliqué à réaliser. Il a bien montré l'intérêt d'utiliser la possibilité de créer plusieurs processus pour un même template, car la première solution serait vite devenue illisible si l'on avait augmenté le nombre de personne. Il est donc très important de bien réfléchir à la modélisation.

Ce TP aura permis de  voir les possibilités du logiciel Uppaal et d'améliorer nos connaissances générales sur les automates. La seule partie du cours que l'on n'aura pas mis en pratique est celle sur les automates probabilistes, peut-être est-il possible de d'améliorer l'exercice en y incluant des probabilités comme par exemple, celle d'appeler quelqu'un ou ne pas appeler.