Rapport technique

Dans ce rapport je vais expliquer le modèle de notre framework et aussi comment modéliser et vérifier les propriétés avec le model ckecker UPPAAL.

1 Modélisation

Dans UPPAAL il existe trois parties de déclaration : *Globale*, *locale*, *système*, ils ont présenté dans la figure suivante.

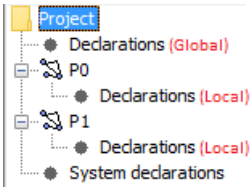

Figure 1 – La partie de déclaration des variable.

Dans cette section je vais présenter la déclaration globale et locale.

1.1 Déclaration globale

Pour déclarer une variable globale, il faut aller vers l'ongle *Editor*, *Project* puis *Declarations*. Figure 2 représente la partie de déclaration des variables.
Ci-dessous l'explication de chaque rôle de variables :

- *bool check* : renvoie True si la signature de message est correcte sinon False.
- *int publicKey, privateKey* : comporte la clé publique et privée de client, de serveur ou de l'attaquant.
- *chan create* : variable de type channel pour communiquer avec le behavior de client et générer la commande.
- *chan c2s,s2c* : canaux de communication entre le client et le serveur.
- *chan sign,signAndCrypt,decrypt,checkSign* : sont des canaux de communication pour communiquer avec l'automate SecureData.
- *msg* : est une variable de type structure de données se compose de variables, fonctions et valeurs.
- *bool System[5]* : est un tableau représente les variables du système sachant que l'indice de tableau c'est la variable et le contenu c'est la valeur de variable.

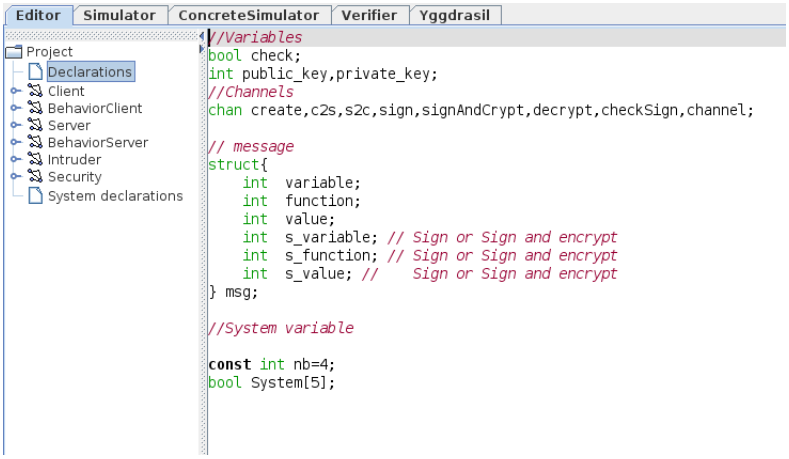

FIGURE 2 – La partie de déclaration des variable.

1.2 Déclaration locale

1.2.1 Client

Ci-dessous l'explication de chaque rôle de variables :

— *bool Csystem[5]* : L'état de variables du système.
— *Update()* : Mise à jour de variables du système.

1.2.2 Serveur

Ci-dessous l'explication de chaque rôle de variables :

— *bool write* : si write est true alors c'est une commande d'écriture sinon c'est une commande de lécture.
— *data* : contient le message de la commande.
— *CheckMsg()* : vérifier le type de commande (écriture ou lecture).
— *Update()* : créer une réponse sur la requêtte de lecture de client.

Modélisation Pour ajouter un nouvel automate il faut aller vers menu *Edit* et aprés sélectionner *Insert Template*. Après l'ajouter de template (automate) il faut la déclarer dans la partie *System Declarations*. Figure 4 représente la partie de déclaration des variables.

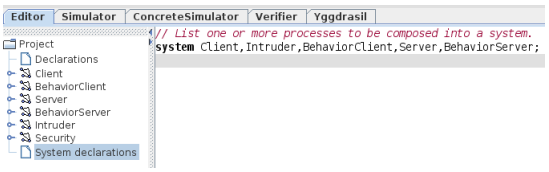

FIGURE 3 – La partie de déclaration des templates.

Pour modéliser un automate on utilise les états et les transitions qui sont présentées dans la figure suivante

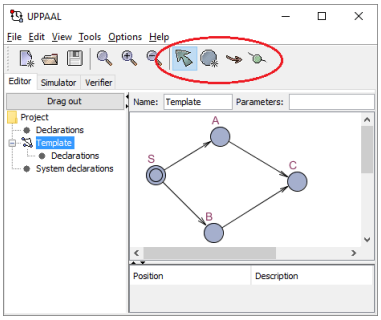

FIGURE 4 – Modélisation d'un automate.

2 Spécifications

Pour vérifier une propriété, il faut aller vers l'ongle *Verifier*, après cliquer sur *Insert* pour ajouter une nouvelle propriété, puis cliquer sur *Check* pour vérifier si le modèle satisfait la propriété ou bien non.

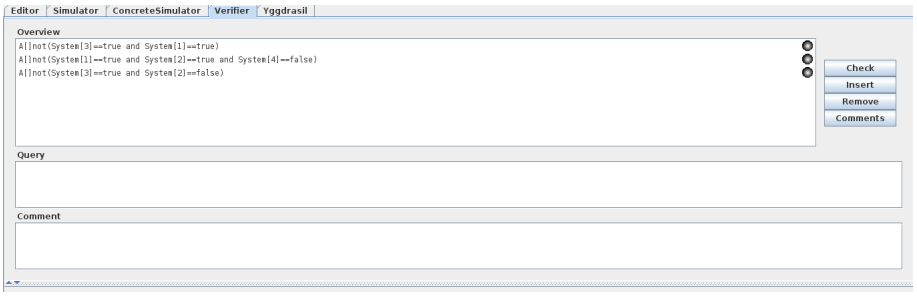

FIGURE 5 – Partie de vérification des propriétés.

Dans la figure 6 la couleur rouge montre que la propriété est non satisfaite

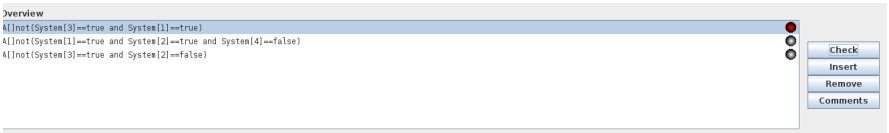

FIGURE 6 – Propriété non satisfaite.

Dans la figure 7 la couleur verte montre que la propriété est satisfaite.

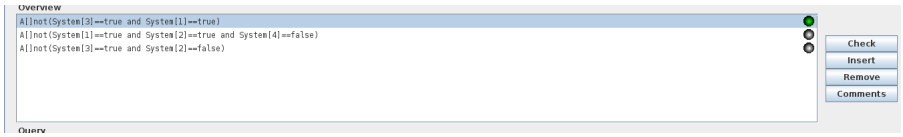

FIGURE 7 – Propriété satisfaite.

3 Comment changer le Behaviors

D'abord il faut modifier le tableau *System* qui représente les variables du système. La taille de tableau représente le nombres des variables de systèmes.

3.1 Client

Le behavior de client est représenté dans l'automate *BehaviorClient* qui utilise la canal *create* pour communiquer avec le client et la variable *msg*.

3.2 Serveur

Le behavior de serveur est représenté dans l'automate *BehaviorServeur* qui utilise la canal *channel* pour communiquer avec le serveur et peux modifier le variable *System*.

4 Comment développer GUI pour les modèles

L'architecture de UPPAAL est client-serveur, le serveur c'est le model checker et le client c'est l'éditeur qui permet de modéliser et vérifier les propriétés. Avec *API model.jar* qui se trouve dans le dossier *lib* de UPPAAL on peut développer un autre client avec langage *JAVA* qui permet de manipuler les modèles automatiquement. Aussi on peut décomplie le fichier JAR d'Uppaal pour utiliser le code source.

lien de documentation : `http://people.cs.aau.dk/~marius/modeldoc/`

5 Exécuter le démo ISSA17

Les étapes suivantes montre comment éxécuter un modèle sous UPPAAL
- Démarrer UPPAAL avec la commande suivante : ***java -jar uppaal***.
- Importer le modèle avec : ***File -> Open system***.
- Aller vers ***Options -> Diagnostic Trace -> (Some ou Shortest ou Fastest)*** pour afficher les traces de contre-exemple.
- Aller vers ***Options -> State Space Representation -> Compact Data Structure*** (la meilleur présentation pour cette exemple).
- Aller vers Verifier et vérifier les propriétés.
- Aller vers Simulator pour voir le contre-exemple (note : UPPAAL affiche un seul contre exemple à la fois).

6 Liste des documents sur le GIT

Dans le dossier stage sur le Git :
— *Dossier biblio :* les travaux similaires de graphe d'attaque.
— *Dossiers experimentation et experimentation-2 :* les résultats de l'expérimentation.
— *Dossier modeles attaquants, modeles attaquants2 :* les rapports du modèle d'attaquants.
— *experimentation-UPPAAl :* Les modèles de UPPAAL.
— *rapport1 :* rapport sur les scénarios d'attaques.
— *rapport2 :* une partie de l'article ISSA 17.
— *Demo_ISSA17 :* l'expérimentation de l'article ISSA 17.
— *Modele_generale :* le modèle générale de l'approche.
— uppaal64-4.1.19 : la version de UPPAAL.