Institut Galilée

Sujet de projet M1

Vérification de systèmes temps-réel distribuée sur un cluster

Clients : Camille COTI et Étienne ANDRÉ
Courriel : {prenom.nom}@lipn.fr

Contexte

Les systèmes temporisés sont devenus omniprésents ces dernières années. Certains d'entre eux (pilotage automatique des avions et drones, voitures sans chauffeur, systèmes bancaires) sont critiques en ce sens qu'aucune erreur ne doit survenir. Tester de tels systèmes peut éventuellement détecter la présence de bugs, mais pas en garantir l'absence. Il est alors nécessaire d'utiliser des méthodes telles que la vérification de modèle (ou *model checking* [BK08]) afin de prouver formellement la correction d'un système. Cette technique représente le système (par exemple un drone ou un protocole de communication) par un modèle, c'est-à-dire une représentation abstraite (et parfois simplifiée) du système, et considère une propriété du système (par exemple la communication va nécessairement se faire en un temps maximal T), et garantit formellement que cette propriété est valide pour le système considéré.

Les systèmes temporisés sont caractérisés par un ensemble de constantes temporelles (ou *paramètres*), telles que la période de lecture d'un capteur d'altitude sur un drone, le temps de traversée d'un circuit par le courant électrique, ou le délai avant la retransmission des données dans un téléphone portable. Des techniques permettant de vérifier le système pour *un* ensemble de constantes existent (et notamment le modèle des automates temporisés [AD94]), et le logiciel UPPAAL [LPY97] permet en particulier une vérification efficace. En revanche, vérifier formellement le système *pour de nombreuses valeurs de ces constantes* peut demander un temps extrêmement long, puisqu'il faut faire cette vérification pour chaque valeur des constantes.

Objectif du projet

Il est intéressant de distribuer cette vérification sur un cluster : pour chaque paramètre, pour chaque valeur entière dans un intervalle prédéfini, on va demander à un nœud du cluster de lancer le logiciel de vérification UPPAAL, et récupérer le résultat. On va ainsi obtenir une cartographie (en n dimensions, où n est le nombre de paramètres) de « bons » points (valeurs des paramètres pour lesquelles le système est correct, c'est-à-dire la propriété est valide) et de « mauvais » points

(valeurs des paramètres pour lesquelles la propriété est invalide). Une sortie graphique avec un logiciel de génération graphique serait intéressante.

Dans un deuxième temps, l'objectif est de s'intéresser à des systèmes particuliers, où augmenter la valeur d'un paramètre pour lequel le système est déjà incorrect va garder le système incorrect ; de façon duale, diminuer la valeur d'un paramètre pour lequel le système est déjà correct va garder le système correct. Dans ce cas, il est inutile de vérifier tous les points : il faut trouver la frontière (où *front de Pareto*) entre les points corrects et incorrects. Voir l'exemple de la Figure 1.

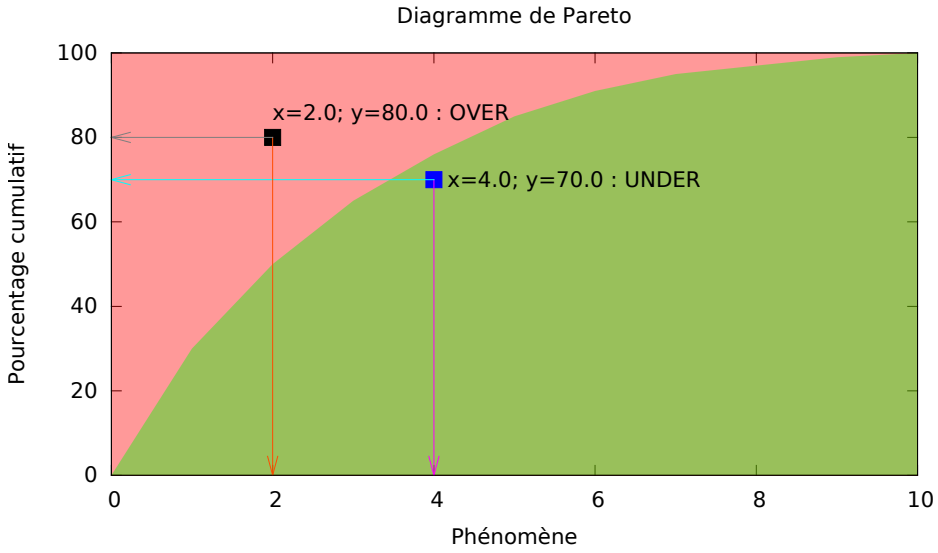

FIGURE 1 – Exemple de front de Pareto à deux paramètres

Objectifs détaillés

Les étapes du projet sont les suivantes :

1. Installation de l'outil UPPAAL, vérification de quelques systèmes temps-réels prédéfinis.
2. Compréhension de la façon d'exécuter UPPAAL depuis un programme externe (script) : modification du fichier du modèle (pour changer la valeur des paramètres), appel de l'outil, et récupération du résultat.
3. Proposition d'un algorithme de distribution de la vérification sur un cluster. Une solution est d'utiliser un algorithme « maître-esclave », où un nœud est responsable de distribuer les points, tandis que les autres se content d'effectuer le travail demandé par le maître.
4. Implémentation de l'algorithme dans un nouveau logiciel, en utilisant un paradigme de programmation distribuée (par exemple MPI). Expérimentation sur un cluster de Sorbonne Paris Cité avec plusieurs dizaines (voire centaines) de processeurs.

5. Proposition d'un second algorithme de distribution de la vérification sur un cluster pour les systèmes tels que la frontière entre les « bons » et les « mauvais » points représente un front de Pareto. On pourra s'inspirer des algorithmes proposés récemment dans [ACN15].
6. Implémentation de ce second algorithme.
7. Génération des sorties graphiques.

Mots-clés

Algorithmique distribuée, méthodes formelles, model-checking, systèmes multiprocesseurs, systèmes temps-réel, synthèse de paramètres

Références

[ACN15] Étienne André, Camille Coti, and Hoang Gia Nguyen. Enhanced distributed behavioral cartography of parametric timed automata. In Michael Butler, Sylvain Conchon, and Fatiha Zaïdi, editors, *Proceedings of the 17th International Conference on Formal Engineering Methods (ICFEM'15)*, Lecture Notes in Computer Science. Springer, November 2015.

[AD94] Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2) :183–235, April 1994.

[BK08] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. MIT Press, 2008.

[LPY97] Kim Guldstrand Larsen, Paul Pettersson, and Wang Yi. UPPAAL in a nutshell. *International Journal on Software Tools for Technology Transfer*, 1(1-2) :134–152, 1997.