# Modélisation et vérification de programmes concurrents récursifs

**Laboratoire :** LIAFA (CNRS, Université Paris7)

**Responsables :** Mihaela Sighireanu et Tayssir Touili

**Contact :** `{sighirea,touili}@liafa.jussieu.fr`

**Contexte du sujet :** Les programmes concurrents se retrouvent actuellement dans nombreaux système informatiques. Etant donnée leur complexité, il est important pour les concepteurs de programmes informatiques de disposer de méthodes rigoureuses qui leur permettent de s'assurer que leurs programmes concurrents remplissent bien les fonctions qui leur sont attribuées, et de détecter les éventuelles erreurs de leurs programmes avant de les mettre en service.

Le problème de la vérification consiste alors à s'assurer qu'un système remplit bien les fonctions qui lui sont attribuées. Ces dernières années, des méthodes de vérification automatiques ont été développées. Ces méthodes concernent essentiellement les systèmes ayant un nombre fini de configurations. Seulement, le nombre de configurations des programmes concurrents est en général arbitaire (non borné). En effet, si on prend un programme concurrent Java dans lequel un thread contrôlant le code X peut lancer en parallèle un autre thread contrôlant X, il est clair que le nombre de threads qui contrôlent X lancés est arbitraire.

Le défi consiste alors à pouvoir vérifier automatiquement ce genre de programmes, où récursion et dynamisme sont présents. Pour ce faire, il nous faut :

- Trouver un modèle mathématique qui soit assez expressif pour pouvoir modéliser récursion, parallélisme, création dynamique de processus, et communication entre ces processus concurrents.
- Trouver des algorithmes d'analyse de ce modèle.

Dans un travail récent [BT03], on propose un modèle appelé PRS permettant de modéliser ces aspects des programmes, ainsi qu'un algorithme d'analyse d'accessibilité de ce modèle. Ce modèle est basé sur des règles de réécriture et l'algorithme d'analyse utilise les automates d'arbre [CDG$^+$97] pour représenter les configurations du programme.

**Résumé du travail souhaité :** Le but du stage est de concevoir un outil d'analyse pour les programmes concurrents en utilisants le travail cidessus. Plus précisément, il s'agit :

1. comprendre le modèle PRS,
2. comprendre l'algorithme d'analyse de PRS,
3. implémenter un outil basé sur cet algorithme et la librairie d'automates d'arbres Timbuk [Ga03],
4. tester ce prototype sur des modèles obtenus à partir de programmes concurrents et, enfin,
5. automatiser cette dernière étape en implémentant un prototype de traduction.

**Résultats escomptés, leur rélévance et leur originalité :** L'outil d'analyse des programmes concurrents qui devrait être le résultat de ce stage serait le premier outil pour ce type de modèle. Il premettra d'étudier les performances de l'algorithme proposé. Le travail d'écriture de programmes concurrents et de traduction vers un modèle PRS sera très util pour étudier les limitations d'expressivité de ce modèle.

**Connaissances préalables attendues chez le stagiaire :** Automates, programmation concurrente, OCAML.

**Bibliographie :**


[BT03] A. Bouajjani and T. Touili. Reachability analysis of process rewrite systems. In *Proceedings of FSTTCD'03*, volume 2803 of *Lect. Notes Comp. Sci.*, pages 74–87. Springer-Verlag, 2003.

[CDG+97] H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. Available on : `http ://www.grappa.univ-lille3.fr/tata`, 1997. release October, 1rst 2002.

[Ga03] T. Genet and all. *Timbuk : A Tree Automata Library.* IRISA, Rennes, France, 2003. Available on `http ://www.irisa.fr/lande/genet/timbuk/`.