Expérimentation, programmation, étude et formalisation de machines d'exécution pour des programmes SugarCubesJS

Directeur de thèse : Pierre Cubaud

Co-encadrants: Olivier Pons et Jean Ferdinand Susini

Description du sujet de thèse :

SugarCubesJS[SCJS]**,** que nous développons au Laboratoire CEDRIC dans l'équipe SyS, est un langage de programmation concurrent s'appuyant sur un modèle de calcul issu de l'approche réactive/synchrone introduite par F. BOUSSINOT dans les années 90[FB96RP]. Ce formalisme propose une décomposition de systèmes en un grand nombre de modules très simples s'exécutant en parallèle. Le comportement de ces composants parallèles est décomposé en étapes successives appelés instants logiques. Les composants parallèles sont alors fortement synchronisés et suivent une horloge logique commune (ils partagent les mêmes instants). A chaque instant, tous ces composants progressent d'une étape. Ces composants communiquent entre eux par diffusion instantanée d'événements. L'intérêt de cette approche est que le modèle de calcul se charge de résoudre efficacement la combinatoire massive générée par ce grand nombre d'échanges. Cela rend l'écriture de systèmes beaucoup plus simple, en déchargeant le programmeur de l'écriture de toute cette combinatoire.
Concrètement, le langage **SugarCubesJS** développé initialement dans le cadre du projet Danse-Doigts[GPST16] est fourni sous la forme d'une bibliothèque de programmation écrite en *Javascript*. Les programmes **SugarCubesJS** ne sont donc pas définis à partir d'une syntaxe textuelle, mais à partir de la construction d'un arbre de syntaxe abstraite, à l'aide d'objets *Javascript*. Ces programmes sont ajoutés dans une machine d'exécution réactive et interprétés. **SugarCubesJS** offre donc une implantation de l'approche réactive/synchrone « à la » BOUSSINOT en *Javascript*, interfacée avec l'environnement événementiel du langage hôte, l'environnement d'un navigateur Web, ou celui de l'ecosystème nodejs[node].

Nous proposons au travers d'expérimentations par l'utilisation de **SugarCubesJS** dans différents domaines d'application (web, simulation, moteur de jeu, objets connectés) de mettre en lumière un certain nombre de problématiques liées aux machines d'exécution de programmes réactifs.

Parallèlement, nous proposons d'étudier de façon rigoureuse, en s'appuyant sur des méthodes formelles, la définition d'une machine d'exécution complète de systèmes réactifs construits à partir de programmes **SugarCubesJS**. En effet, si la construction de programmes réactifs est de nos jours bien comprise, ainsi que la définition de ce que peut être l'exécution d'un instant d'un programme, la définition d'un exécutif réactif reste quelque chose d'assez peu étudié dans ce domaine. C'est-à-dire, la façon dont sont interfacés les événements extérieurs au système réactif et les événements **SugarCubesJS**, mais aussi le rythme des réactions (passage des instants successifs) des machines d'exécution.
Classiquement, les machines d'exécution sont mises en œuvre par le développeur de systèmes réactifs de façon ad hoc. Souvent, il suffit d'appeler cycliquement une réaction de la machine réactive et d'intégrer entre deux réactions consécutives les signaux de l'environnement d'exécution sous la forme d'événements réactifs qui seront traités par les programmes au cours d'un instant du système.
Cependant, les expérimentations récentes sur le *Web* ont révélé la difficulté particulière de mettre en œuvre cette approche dans le domaine du *Web*[ElSibaie18]. Comment les événements extérieurs doivent être pris en compte et à quel rythme ? Ces questions rendent complexe la mise en œuvre de la machine d'exécution. Ceci est d'autant plus complexe que les points d'activation *Javascript* sont nombreux au sein d'un même document et disposent de contraintes spécifiques.

Ces contraintes nouvelles ne peuvent pas être évacuées dans la définition simple de l'exécutif, comme cela est classiquement le cas dans le monde synchrone. En effet, certaines contraintes imposent par exemple des réactions instantanées (contraintes d'interactivité dans les navigateurs Web qui imposent que certaines actions soient directement la conséquence d'une sollicitation explicite de l'utilisateur). D'autres sont des contraintes de performance ou d'économie d'énergie, ou encore des contraintes de sécurité. Dans ces conditions, l'écriture d'un exécutif réactif devient plus complexe et surtout n'est plus orthogonale à l'écriture du système réactif lui-même. Dans le cadre de cette thèse nous proposons de réfléchir sur des éléments de spécification de l'exécutif réactif ajoutés à la définition de programmes, afins de tenir compte de ces contraintes sans pour autant complexifier la définition de programmes réactifs. Des annotations ou de nouvelles constructions syntaxiques seront peut-être envisagées.

Nous proposons également d'expérimenter cette approche dans la simulation scientifique et la simulation de mondes virtuels qui sont deux terrains d'application privilégiés de cette approche de la programmation[BMS14]. Mais plus généralement, nous proposons de généraliser nos propositions à la description formelle de contraintes de différents modes d'interactions, y compris par exemple dans le domaine de l'informatique embarquée.

Enfin, définir la sémantique formelle de ce modèle de calcul et apporter la preuve de certaines propriétés escomptées est une tâche complexe. Vérifier que l'implantation de ce modèle est conforme aux règles de la spécification formelle est également un problème que nous souhaitons aborder. Une première étape pourra être de prouver des propriétés de terminaison

et de confluence à l'aide d'outils de réécriture et de preuve de terminaison comme Cime [CCFP11]. Puis de poursuivre vers une extraction automatique d'une implantation de référence dans un assistant de preuve comme Coq[Coq19].

Références

[FB96RP] Frédéric BOUSSINOT. *La programmation réactive - Application aux systèmes communicants, Collection technique et scientifique des télécommunications*. MASSON, 1996.

[Coq19] The coq proof assistant, 1984. http://coq.inria.fr/ (last checked 2019).

[ElSibaie18] R. El Sibaie : "Programmation Web Réactive dans un cadre typé statiquement pour l'orchestration de contenus mutltimédia riches", thèse, soutenance 12/07/2018, direction de recherche Chailloux, Emmanuel (2018).

[BMS14] Boussinot, Frédéric & Monasse, Bernard & Susini, J.-F. (2014). Reactive Programming of Simulations in Physics. International Journal of Modern Physics C. 26. 10.1142/S0129183115501326.

[GPST16] Susini, J.-F & Pons, Olivier & Guedin, Nolwenn & Thevenot, Catherine. (2016). Danse-doigts, jeu de motricité fine.

[CCFP11]Contejean, E., Courtieu, P., Forest, J., Pons, O., & Urbain, X. (2011). Automated certified proofs with CiME3. In *22nd International Conference on Rewriting Techniques and Applications (RTA'11)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[node]https://nodejs.org/en/
[SCJS]https://github.com/LordManta/SugarCubesJS