INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Projet MIMOSA

Migration et Mobilité : Sémantique et Applications

Sophia Antipolis

THÈME 1C

Table des matières

MIMOSA *est un projet commun entre l'INRIA, le Centre de Mathématiques Appliquées (CMA) de l'Ecole des Mines de Paris et le Centre de Mathématiques et d'Informatique (CMI) de l'Université de Provence.*

1 Composition de l'équipe

Responsable Scientifique
Gérard Boudol [Directeur de Recherche, Inria]

Responsable Permanent
Ilaria Castellani [Chargée de Recherche, Inria]

Assistantes de Projet
Catherine Juncker [Inria]
Dominique Micollier [Armines]

Personnel Inria
Davide Sangiorgi [Directeur de Recherche, Inria]
Manuel Serrano [Chargé de Recherche, Inria, à partir d'octobre]

Personnel CMA et CMI
Roberto Amadio [Professeur, université de Provence]
Frédéric Boussinot [Maître de Recherches, École des Mines de Paris]
Silvano Dal-Zilio [Chargé de Recherche, CNRS, à partir d'octobre]
Jean-Ferdy Susini [Ingénieur Armines, à partir d'octobre]

Chercheurs Doctorants
Raul Acosta-Bermejo [Allocataire CONACYT, 1ère année]
Christian Brunette [Allocataire MENRT, à partir d'octobre]
Cédric Lhoussaine [ATER, université de Provence, 3ème année]
Charles Meyssonnier [Elève ENS Lyon, à partir de septembre]
Dimitri Petoukhov [Allocataire École des Mines de Paris, à partir de septembre]
Jean-Ferdy Susini [Allocataire École des Mines de Paris, jusqu'à octobre]
Vincent Vanackere [Allocataire ENS Lyon, 1ère année]
Pascal Zimmer [Allocataire ENS Lyon, 1ère année]

Chercheurs Post-Doctorants
Josva Kleist [Projet RNRT MARVEL, jusqu'à août]
Antonio Ravara [Projet RNRT MARVEL, jusqu'à avril]

Stagiaires
Julien Demaria [DEA Informatique, Nice]
Christian Brunette [DEA RSD, Nice]
Etienne Lozes [DEA DIF, ENS Lyon]
Charles Meyssonnier [DEA DIF, ENS Lyon]
David Teller [DEA DIF, ENS Lyon]

2 Présentation et objectifs généraux

Le projet MIMOSA est commun avec le Centre de Mathématiques Appliquées de l'ENSMP et le Centre de Mathématiques et d'Informatique de l'université de Provence. L'objectif général du projet est de concevoir et d'étudier des modèles de la programmation répartie et mobile, d'en tirer des primitives pour cette programmation, et d'élaborer des techniques de raisonnement et de vérification concernant spécifiquement les problèmes liés au code migrant. Nous développons deux approches : l'approche interactive, fondée sur l'idée de processus indépendants et communicants, et l'approche réactive, où les processus réagissent de concert à des événements diffusés. Nous visons particulièrement à intégrer les primitives de migration dans la programmation réactive, de façon à pouvoir programmer des «agents» participant à des «assemblées réactives» (jeux en réseau par exemple). Nos axes de recherche principaux sont les suivants :

- Modèles et langages pour la mobilité, approche interactive. Dans cet axe de recherche l'objectif est d'étudier les primitives pour la mobilité fondées en particulier sur le modèle du pi-calcul, et de ses extensions réparties.
- Modèles et langages pour la mobilité, approche réactive. Les objectifs sont de poursuivre le développement des outils de la programmation réactive et d'y intégrer des primitives pour la migration.
- Méthodes de raisonnement. Nous développons de telles méthodes pour les modèles de la migration, comme le calcul des Ambients, fondées sur la bisimulation et les logiques temporelles, ainsi que des méthodes de vérification pour des problèmes de sécurité.
- Types. Les systèmes de types sont utilisés, dans les modèles de la mobilité, à la fois pour éviter des erreurs, imposer des disciplines de sécurité, spécifier des comportements, mais aussi pour valider des transformations et des équivalences.

Relations internationales et industrielles :

- Relations contractuelles bilatérales avec l'université de Lisbonne, sur la sémantique des objets concurrents, et avec l'université de Pennsylvanie, sur l'utilisation des types dans le raisonnement à propos des systèmes concurrents.
- Projet RNRT MARVEL, avec France Télécom R&D et l'ENST, sur la définition d'un modèle de programmation répartie avec objets mobiles, et d'une machine virtuelle associée.
- Deux CTI France Télécom R&D, l'une sur les objets réactifs distribués, l'autre sur la modélisation et la vérification des objets migrants.
- Projet IST PING, regroupant sept partenaires, où nous intervenons sur l'utilisation des techniques réactives dans la programmation des jeux en réseau.

3 Fondements scientifiques

3.1 Sémantique de la mobilité et sécurité

Mots clés : concurrence, sémantique, parallélisme asynchrone, typage, langages fonctionnels, objets concurrents, mobilité, λ-calcul, π-calcul, protocoles cryptographiques,

non-interférence.

Participants : Roberto Amadio, Gérard Boudol, Ilaria Castellani, Cédric Lhoussaine, Davide Sangiorgi, Vincent Vanackere.

Résumé : *La* mobilité *est devenue l'un des aspects importants des systèmes informatiques, et particulièrement des systèmes distribués. Notre projet s'intéresse à la notion de «code mobile» – et donc à une notion de mobilité logique, plutôt que physique. Il s'agit de dégager les concepts utilisés dans les langages de programmation récemment proposés pour permettre la mobilité des calculs et d'en formaliser la sémantique. D'autre part, les préoccupations concernant la* sécurité *sont évidemment très importantes s'agissant de code mobile, et plus généralement de systèmes partagés. Nous étudions des méthodes de vérification de propriétés de sécurité.*

Les modèles sur lesquels nous développons principalement notre approche formelle de la mobilité sont le π-*calcul* de Robin Milner et le calcul des «*Ambients*» de Luca Cardelli. Le premier est un modèle similaire à celui que le λ-calcul offre pour la programmation séquentielle et fonctionnelle, mais dans lequel le parallélisme est pris en compte, ainsi qu'une certaine forme de mobilité puisque dans le π-calcul les processus se communiquent des noms de canaux de communication. En fait, le π-calcul contient plus ou moins directement le λ-calcul, sa puissance d'expression est donc déjà bien établie de ce simple fait – ceci en regard de la programmation séquentielle.

Le second explore plutôt le concept de migration : il est fondé sur une notion très générale de «domaine», appelé «Ambient», dans lequel des calculs peuvent s'effectuer, et qui possède une structure hiérarchique. Dans le calcul des «Ambients», la structure d'imbrication des domaines les uns dans les autres évolue dynamiquement, et tout le calcul réside dans ce mouvement (et aussi dans la possible disparition des frontières). Ce mécanisme est en fait extrêmement général, puisqu'on peut l'utiliser pour simuler les machines de Turing.

Nos recherches sur ces formalismes, considérés en tant que modèles pour la programmation des systèmes distribués et mobiles, s'articulent autour de plusieurs axes. Nous étudions en particulier comment peut s'étendre la notion de type, et comment elle peut être utile dans le raisonnement à propos des programmes répartis ou migrants. Cette notion de type, familière dans la programmation séquentielle, paraît encore plus importante dans un cadre distribué, parce que les sources d'erreurs y sont bien plus nombreuses et subtiles. Plus généralement, nous étudions ce que peuvent être des méthodes de raisonnement relatives aux programmes répartis et migrants. Nous visons à appliquer ces méthodes tout spécialement aux problèmes de sécurité, par exemple dans la vérification de protocoles cryptographiques ou l'analyse de flux d'information dans les programmes.

3.2 Programmation réactive

Participants : Raul Acosta-Bermejo, Frédéric Boussinot, Jean-Ferdy Susini.

Mots clés : réactif, objet, script, parallélisme, événement, World Wide Web, Reactive-C, Java, SugarCubes, Icobj, Junior, Rejo, FairThreads, distribution, migration.

Résumé : *L'approche réactive a pour but d'étudier divers modèles de programmation dans lesquels existe une horloge logique définissant des instants globaux. Elle a également pour objectif d'implémenter des langages de programmation construits sur ces modèles.*

Le traitement de la concurrence dans les langages de programmation usuels reste très empirique et fondé sur des concepts datés. Par exemple, JAVA propose les «threads» et les verrous pour la gestion de la concurrence. Nos travaux sur la programmation réactive, initiés avec le langage Reactive-C en 1988, visent à ajouter aux langages existants des primitives de programmation de haut niveau, fondées sur les paradigmes de la programmation synchrone : notion d'instant global, d'événement et de diffusion instantanée.

Plusieurs langages et formalismes réactifs ont été définis et implémentés en Reactive-C : réseaux de processus réactifs, langage synchrone SL, objets réactifs. Ils sont décrits dans un livre qui présente également Reactive-C en détail [4]. L'approche réactive au dessus de JAVA (plus particulièrement les SugarCubes) est décrite dans un livre [5].

Les SugarCubes sont un ensemble de classes JAVA pour la programmation réactive. Les SugarCubes proposent une communication de haut niveau à base d'événements diffusés instantanément, dans laquelle la réaction à l'absence est reportée à l'instant suivant. Les Scripts Réactifs apportent toute la souplesse de l'interprétation à l'approche réactive. La présence des instants permet de définir une migration dans des états cohérents des scripts réactifs à travers le réseau. Nous avons implémenté en SugarCubes une version des scripts réactifs au dessus de JAVA. L'objectif de la programmation par Icobjs est de fournir une technique de programmation entièrement graphique, à base de combinaisons de comportements. Les comportements sont en fait des scripts réactifs, combinés de différentes façons par des manipulations d'icônes à l'écran.

Le formalisme Junior récemment introduit est un noyau de primitives pour la programmation réactive au dessus de JAVA. Junior est en quelque sorte le noyau des SugarCubes. Trois objectifs ont motivé la création de ce formalisme : prendre en compte le parallélisme asynchrone des systèmes distribués ; donner une sémantique formelle aux primitives réactives ; enfin, réaliser des implémentations efficaces et supportant un grand nombre d'événements diffusés.

Nous avons commencé à appliquer la programmation réactive à la conception de mondes virtuels, dans le cadre du projet européen PING. Ce projet cible plus particulièrement les jeux en réseau avec grand nombre de joueurs. Dans ce projet, nous utilisons Junior et les Icobjs pour les comportements d'objets virtuels.

4 Domaines d'applications

4.1 Télécommunications

Mots clés : télécommunications, programmation réactive, mobilité.

Le domaine d'applications «naturel» de nos travaux est celui des télécommunications. Nos contributions y portent à la fois sur la spécification et sur la programmation fiable à l'aide de formalismes adaptés de haut niveau.

4.1.1 Modélisation de systèmes mobiles

La spécification *formelle* des systèmes répartis, par exemple exprimés comme systèmes d'objets concurrents, reste encore très largement à faire, et les besoins dans ce domaine sont encore mal couverts. Nos travaux dans ce domaine ont fait l'objet d'une collaboration avec France Télécom R&D, dans le cadre de deux CTI successives, et se poursuivent actuellement au sein du projet RNRT MARVEL, qui vise concrètement la conception d'une machine virtuelle répartie pour la mobilité.

4.1.2 Plates-formes distribuées d'exécution réactive

Un besoin existe de plates-formes d'exécution répartie (DPE) utilisant ou implémentant l'approche réactive. Nous travaillons en collaboration avec France Télécom R&D pour l'étude et la réalisation de ce type d'infrastructure d'exécution. Un des objectifs est la mise en place de mécanismes liés à la «qualité de service» (QoS) et utilisant l'approche réactive. Un second objectif est le «passage à l'échelle» de systèmes dans lesquels intervient un très grand nombre de composants parallèles et d'événements diffusés.

4.2 Multimédia

4.2.1 IHM graphiques

L'utilisation des Icobjs apparaît naturelle pour les interfaces homme-machine télécom, en particulier lorsqu'il s'agit de représenter des animations d'icônes, comme par exemple dans la supervision de réseau. Une autre utilisation possible des Icobjs est la conception de systèmes permettant à l'utilisateur final de programmer ses propres services, par des combinaisons graphiques d'Icobjs. Enfin, le domaine des jeux, en particulier des jeux en réseaux, semble être un domaine dans lequel les Icobjs peuvent être utilisés, par exemple pour la programmation des avatars de joueurs.

5 Logiciels

5.1 Programmation réactive

Participants : Raul Acosta, Frédéric Boussinot [correspondant], Jean-Ferdy Susini.

Mots clés : Programmation réactive, Java, Reactive-C, SugarCubes, Junior, Rejo, Fair Threads.

Résumé : *L'ensemble des logiciels développés autour de l'approche réactive est disponible librement sur le Web. Cet ensemble va de Reactive-C aux SugarCubes en passant par Junior, Rejo et les FairThreads. Tous ces logiciels sont disponibles sur le Web en* `http://www.inria.fr/mimosa/rp/downloads`.

Reactive-C Reactive-C est un préprocesseur pour une programmation réactive en C. Reactive-C a été utilisé comme «assembleur réactif» pour implémenter plusieurs formalismes réactifs comme le langage synchrone SL, les scripts réactifs, ou la programmation par icobjs.

SugarCubes SugarCubes est un ensemble de classes 100% JAVA pour une programmation réactive en JAVA. La programmation par icobjs en JAVA a été implémentée avec les SugarCubes. Une comparaison entre les threads JAVA et les SugarCubes est disponible en `http://www.inria.fr/mimosa/rp/SugarCubes/ComparisonThreadsSugarCubes`.

Junior Junior est une API pour une programmation réactive en JAVA. Junior peut être vu comme le noyau des SugarCubes. Il possède plusieurs implémentations, dont certaines destinées à traiter un grand nombre de composants parallèles.

Rejo Rejo est un langage qui introduit des objets réactifs au dessus de JAVA. Rejo dispose de primitives pour la migration de code à travers le réseau. Les programmes Rejo sont compilés en Junior et exécutés sur une plate-forme dediée appelée ROS.

FairThreads Les fair threads sont des threads coopératifs exécutés par un ordonnanceur équitable qui leur donne un accès égal au processeur. Ils possèdent une sémantique formelle ainsi qu'une implémentation efficace en JAVA. Les fair threads sont reliés à l'approche réactive (Junior).

5.2 Typage des mixins

Participants : Pascal Zimmer [correspondant].

Mots clés : inférence de type, récursion, objets, mixins.

Nous avons implémenté, en OCAML, une extension de l'algorithme de typage pour la récursion généralisée dans un langage à la ML décrit dans [12, 19]. L'extension par rapport aux travaux cités concerne aussi bien le langage de base, qui contient des constructions standards sur les booléens, les entiers, les listes, etc., que le système de type, qui autorise des types récursifs. Cette implémentation permet l'expérimentation d'un style de programmation orientée-objet basé sur les *mixins*.

5.3 Implémentation des Ambients

Participants : Davide Sangiorgi [correspondant].

Une implémentation d'un langage de programmation expérimental basé sur les «safe Ambients» de Levi et Sangiorgi ([1]) est en cours de développement. Un prototype devrait être disponible d'ici la fin de l'année. Ceci est un travail en collaboration avec Andrea Valente de l'université de Turin [14].

[1] voir le rapport de l'an passé

6 Résultats nouveaux

6.1 Sémantique de la mobilité

Participants : Roberto Amadio, Cédric Lhoussaine, Davide Sangiorgi.

6.1.1 Travaux sur le π-calcul

Davide Sangiorgi a terminé son ouvrage, écrit en collaboration avec David Walker, sur le π-calcul. Ce livre est maintenant publié [7]. Il couvre l'ensemble de la théorie du π-calcul, depuis les bases (syntaxe, sémantique opérationnelle, équivalences comportementales) jusqu'au codages des fonctions et des objets, en passant par les sytèmes de types et leur utilisation pour le raisonnement.

L'écriture de ce livre a conduit Sangiorgi à de nouveaux résultats concernant les équivalences dans le π-calcul, publiés dans [15]. Plus précisément, il est montré que si les sommes arbitraires sont admises dans le π-calcul, alors on peut prouver facilement que les congruences «à barbes» et «précoce» coincident. Une autre équivalence, la bisimilarité «à barbes» ouverte est introduite, dont il est montré qu'elle ne correspond à aucune bisimulation connue. Une caractérisation, en termes de bisimulation étiquetée, en est donnée.

Cédric Lhoussaine a poursuivi l'étude du π-calcul distribué réceptif ([2]). Il a en particulier étudié le problème de l'inférence de types pour une version étendue de $\mathrm{D}\pi_1^r$ très proche de $\mathrm{D}\pi$ de Hennessy et Riely. Ce sont des π-calculs distribués comprenant des notions explicites de *localités* et de *migration* où l'espace est *plat* (i.e. il n'y a pas d'imbrications de localités) et la communication est *locale* (i.e. les messages d'une localité à une autre doivent être explicitement «routées»). De plus, les noms de localités sont typés et on utilise une relation de sous-typage explicite sur les types de localités. Cette dernière permet de définir une notion de *type principal*. Lhoussaine décrit ensuite un algorithme d'inférence de types dont il démontre qu'il produit un type principal pour tout terme typable.

Roberto Amadio et Charles Meysonnier se sont intéressés aux problèmes de décision (accessibilité, *model-checking*) dans les modèles qui incluent un mécanisme de génération dynamique de noms. Dans [10] ils étudient la décidabilité d'un problème d'accessibilité pour plusieurs fragments du π-calcul asynchrone. Les combinaisons de trois aspects principaux – la génération de noms, la mobilité de noms et le contrôle non borné – sont considérées. Amadio et Meyssonnier montrent que la combinaison de la génération de noms avec, ou bien la mobilité de noms, ou bien le contrôle non borné induit un fragment indécidable. D'autre part, ils démontrent que la génération de noms sans mobilité de noms et avec contrôle borné est décidable par réduction au problème de couverture pour les réseaux de Petri. Ce travail ouvre un certain nombre de perspectives concernant : l'existence de fragments décidables avec une forme limitée de mobilité, la géneralisation des propriétés décidables, et l'introduction et expérimentation de méthodes d'approximation.

[2] voir les rapports précédents

6.1.2 Travaux sur le calcul des «Ambients»

Le travail sur l'implémentation du calcul des «Safe Ambients» ([3]) a abouti : dans l'article [14] on décrit une machine abstraite pour une implémentation répartie de ce calcul. Dans notre modèle, la topologie de la configuration courante des «Ambients» peut être différente de celle de leur répartition physique. La correction de cette machine par rapport à la sémantique opérationnelle est démontrée. Notre machine abstraite est différente, et beaucoup plus simple que les modèles proposés jusqu'à présent. La raison principale de cette simplicité est que la machine implémente les «Safe Ambients» typés, plutôt que le calcul originel, non typé, des «Ambients».

Le travail de Davide Sangiorgi sur la logique des «Ambients» a été publié [16]. On rappelle que la logique des «Ambients» a été proposée pour exprimer des propriétés de répartition spatiale et d'évolution temporelle dans le calcul des «Ambients» (cette logique se trouve être également applicable aux langages de requètes concernant les données semi-structurées). Afin de comprendre la puissance d'expression de cette logique, on compare l'équivalence – i.e. satisfaire les même formules – qu'elle induit sur les «Ambients» à des équivalences connues, comportementales ou structurelles. Ainsi on obtient une caractérisation de l'équivalence logique sous la forme d'une bisimulation, et une axiomatisation de cette équivalence est donnée pour le cas des processus finis. Le résultat est que, d'une façon assez surprenante – par rapport aux résultats usuels concernant la logique de Hennessy-Milner par exemple –, la logique des «Ambients» permet une observation très fine de la structure interne des processus de ce calcul. Une conséquence de cette étude est une meilleure compréhension de l'équivalence comportementale des «Ambients». Par exemple, nous montrons que cette équivalence masque les comportements d'«aller-retour», comme le fait d'entrer puis de sortir de façon répétée d'un même «Ambient».

Mentionnons ici pour finir que le travail de Pascal Zimmer sur l'expressivité du calcul des «Ambients» décrit dans le rapport de l'an passé a été accepté pour publication dans les journal MSCS [22].

6.2 Sécurité

Participants : Roberto Amadio, Gérard Boudol, Ilaria Castellani, Vincent Vanackere.

6.2.1 Protocoles cryptographiques

Les prototocoles cryptographiques jouent un rôle important dans la sécurisation d'applications réparties. Il existe de nombreuses techniques de spécification des protocoles cryptographiques (et des propriétés qu'ils sont censés vérifier) dans des logiques variées : logique du premier ordre, logique linéaire, algèbres de processus, logique équationnelle... Un certain nombre d'expériences montrent que les spécifications peuvent s'exprimer naturellement dans des fragments de la logique du premier ordre.

Notre objectif est de contribuer à la vérification de ce type de protocoles en utilisant des méthodes symboliques définies sur ces fragments logiques. Dans un premier temps nous

[3] voir le rapport de l'an passé

étudions le problème de l'accessibilité pour les protocoles cryptographiques représentés comme des processus utilisant des fonctions cryptographiques *parfaites* [17]. Ceci nous a conduits à définir un système de réduction symbolique qui est le point de départ de nos travaux sur les protocoles. Ce système fournit une procédure de décision pour les processus *finis* et une référence pour des implémentations correctes. Le système de réduction symbolique peut être considéré comme une variante de l'unification syntaxique qui est compatible avec certaines contraintes ensemblistes. Pour un fragment significatif de notre formalisme, nous proposons une implémentation du système de réduction symbolique basée sur les graphes dirigés acycliques qui mène à un algorithme dont la complexité est en temps polynomial non-déterministe, ce qui correspond à la borne inférieure du problème.

La question des processus *itérés* ou *à contrôle fini* est plus délicate car nous avons montré que le problème est indécidable. Nous travaillons à définir des classes de protocoles suffisamment expressives pour lesquelles les problèmes restent décidables. Nous avons déjà un élément de réponse pour une sous-classe de processus itérés qui n'utilisent pas les couples. Notre technique est basée sur les transductions rationnelles de langages rationnels et elle s'applique à une classe de processus qui contient les protocoles *ping-pong* présentés dans Dolev et al..

6.2.2 Non-interférence

Nous nous sommes également intéressés, en ce qui concerne la sécurité, au problème de la *non-interférence*. Le problème est ici le suivant : étant donnée une «politique de sécurité», qui prend la forme d'un treillis de niveaux de sécurité, il s'agit de trouver des moyens de garantir que l'exécution d'un programme ne peut donner lieu à des «fuites d'information» d'un niveau de sécurité vers un autre qui ne lui est pas supérieur. Par exemple, on veut garantir que deux «applets» partageant des données mais ayant également chacune des données «privées» – donc de niveaux de sécurité incomparables – peuvent coopérer sans qu'il y ait une fuite de données privées de l'une vers l'autre. Ce problème est posé depuis longtemps, mais a trouvé assez récemment une solution très élégante, avec les travaux de Volpano et Smith qui ont formalisé une technique d'analyse due à Denning sous la forme d'un système de types, obtenant ainsi une preuve de correction de cette analyse.

Dans [11], puis dans [18], nous avons raffiné la technique de Volpano et Smith qui, dans le cas de programmes concurrents, était beaucoup trop restrictive. Notre idée a été de tenir compte dans les types, non seulement du niveau de sécurité des variables affectées par un programme – c'était l'idée originale de Volpano et Smith –, mais encore du niveau des variables testées (dans un branchement conditionnel ou une boucle «while»). La contrainte que l'on impose alors est que l'on ne peut affecter une variable que d'un niveau supérieur à celui des variables préalablement testées. Nous montrons que ce système de types assure bien la propriété souhaitée – absence de fuite –, tout en étant moins restrictif que celui de Volpano et Smith. Le même système de types a été – fait remarquable – découvert indépendamment par Smith lui-même, et publié à peu près simultanément. Dans [18], nous étendons ce système de types au cas où des programmes (séquentiels) concurrents sont soumis à un ordonnancement. Nous montrons comment modéliser cette situation, qui en général introduit de nouvelles possibilités de fuites, par de nouvelles primitives de synchronisation, et nous généralisons le résultat montrant que les programmes typables sont sûrs. A notre connaissance, c'est le seul résultat

sur cette question qui n'utilise pas une approche probabiliste, et qui permette de classifier, par typage, les mécanismes d'ordonnancement.

6.3 Typage de la récursion et sémantique des objets

Participants : Gérard Boudol, Pascal Zimmer.

Un effort a été entrepris, notamment dans le cadre des contrats «Sémantique des Objets Concurrents» avec l'université de Lisbonne et «Objets Migrants» avec France Télécom R&D, pour obtenir une formalisation de la programmation orientée objet, particulièrement importante dans le cadre des plates-formes réparties. L'objectif était surtout d'obtenir un modèle autorisant l'inférence de type à la ML – ce qui n'est pas le cas des «calculs d'objets» à la Abadi-Cardelli ou Fisher-Mitchell par exemple. Un candidat naturel est le modèle proposé initialement par Cardelli, et repris par Wand en ce qui concerne le typage, où les concepts de la programmation objet sont dérivés de notions simples et traditionnelles, et où en particulier un objet est vu comme un *enregistrement récursif*. Toutefois, il est nécessaire d'adapter ce modèle à un langage en appel par valeur, pour pouvoir modéliser des objets possédant un état interne. Une difficulté spécifique se pose alors, qui est d'étendre le mécanisme de la récursion de ML, beaucoup trop restrictif – il ne permet, pour l'essentiel, de construire que des fonctions.

Nous avons ainsi proposé [12, 19] une forme d'analyse statique, intégrée au système de types usuel, qui permet de garantir qu'une variable peut être liée récursivement de façon sûre, c'est-à-dire sans risque d'introduire des erreurs à l'exécution. Nous avons montré que les bonnes propriétés du typage à la ML sont préservées, à savoir : pas d'erreur à l'exécution pour les expressions typables, existence d'une procédure de décision concernant la typabilité qui produit un type le plus général possible (type «principal») pour chaque expression typable. Les contraintes à résoudre pour mener à bien l'analyse de sûreté des variables récursives étant très simples, la complexité de l'algorithme d'inférence de type n'est pas affectée. Cet algorithme a été implementé par Pascal Zimmer, pour un système de types qui comprend des types récursifs.

Grâce au typage raffiné de la récursion, on peut alors étendre, dans un langage à la ML comportant des enregistrements extensibles, la modélisation des objets de Cardelli et Wand au cas d'objets avec état. Dans l'article [19], nous montrons comment modéliser les constructions à base de *mixins*, qui sont essentiellement celles proposées par Bracha avec JIGSAW. On obtient un style de programmation très souple car, les «mixins» étant des classes paramétrées par leur super-classe, on peut s'affranchir de la rigidité de la hiérarchie des classes des langages à objets traditionnels (à héritage simple). Notons aussi que notre système de types pour la récursion a déjà été utilisé par Hirschowitz et Leroy pour des constructions de modules mutuellement récursifs dans ML.

6.4 Programmation réactive

Participants : Raul Acosta, Frédéric Boussinot, Jean-Ferdy Susini.

La version 3 des SugarCubes est maintenant disponible sur le Web avec ses sources. Elle introduit de nouvelles constructions augmentant la puissance expressive du langage, en particulier celle d'événement valué, compatible avec le modèle de programmation des cubes. Les

SugarCubes v3 sont décrits en détails dans la thèse de Jean-Ferdy Susini [8].

REJO et ROS [9, 23] sont deux outils expérimentaux basés sur l'Approche Réactive. REJO est une extention à JAVA qui génère des objets qui peuvent être considérés comme des Objets Réactifs, avec leurs données et leurs instructions réactives. Ces objets peuvent être aussi considérés comme des Agents Mobiles car ils peuvent migrer sur une plate-forme, appelée ROS, qui offre les fonctionnalités dont ils ont besoin. REJO et ROS sont construits au-dessus de Junior. Le modèle de programmation de REJO, qui est celui de Junior, fournit des instructions réactives tournant sur une machines réactive. Les instructions réactives implémentent des événements diffusés instantanément, ainsi qu'un opérateur de parallélisme logique qui n'utilise pas de thread.

Nous avons dégagé un noyau de programmation réactive en JAVA que nous avons appelé Junior (Jr, J pour JAVA, r pour réactif), pour lequel nous avons défini une sémantique formelle sous forme de règles de réécriture conditionnelles [8]. Junior peut être vu comme le noyau des SugarCubes. Nous avons donné plusieurs implémentations de Junior, l'une d'entre elles étant capable de traiter un grand nombre de composants parallèles et d'événements (de l'ordre de plusieurs milliers). Nous avons défini une API de programmation en Junior. La version actuelle est la 2.1.

Les *fair threads* [20] sont des threads coopératifs exécutés par un ordonnanceur qui leur distribue le processeur équitablement. Les fair threads peuvent communiquer en utilisant des événements diffusés. Ils sont complètement portables car leur sémantique ne dépend pas de la plate-forme d'exécution. Les fair threads peuvent être contrôlés finement, ce qui permet à l'utilisateur de coder ses propres stratégies d'exécution. Ils sont définis dans le contexte du langage JAVA et sont utilisables au travers d'une API. La sémantique des fair threads est très proche de la programmation réactive dont ils sont issus.

Nous avons fourni une première série de rapports comme «livrables» du projet PING concernant les mondes virtuels. Ces rapports décrivent l'utilisation de la programmation réactives (Junior, Rejo et Icobjs) dans le cadre des mondes virtuels distribués. Un papier présenté à Web3D décrit ce travail [13]. Nous avons également développé plusieurs démos de programmation réactive au dessus de la plate-forme PING [24].

Julien Demaria lors de son stage de DEA encadré par F. Boussinot et M. Serrano a défini une couche réactive au dessus de Bigloo, un système de programmation Scheme [25]. Cette couche appelée Senior a également été implementée lors du stage. Senior est disponible sur le Web en `http://www.inria.fr/mimosa/rp/Senior`.

7 Contrats industriels (nationaux, européens et internationaux)

7.1 Projet IST PING

Participants : Raul Acosta, Frédéric Boussinot, Jean-Ferdy Susini.

Le projet PING, qui a débuté en juin 2000 pour une durée de 2 ans, se propose de réaliser une plateforme pour des mondes virtuels possédant un grand nombre de participants. Les jeux en réseau sont une des cibles du projet. Notre tâche dans le cadre de PING est plus particulièrement de fournir un environnement de programmation réactive pour la conception

des objets peuplant le monde. Nous étudions également comment les techniques réactives peuvent être utilisées pour préserver la cohérence des diverses visions distribuées du monde virtuel.

7.2 Projet RNRT MARVEL

Participants : Roberto Amadio, Gérard Boudol, Ilaria Castellani, Davide Sangiorgi.

Ce projet RNRT, dont l'intitulé complet est «Machine répartie virtuelle et langage pour objets mobiles», d'une durée de 3 ans, a débuté en décembre 1998. Il inclut comme partenaires France Télécom R&D (coordinateur), le projet MOSCOVA de l'INRIA de Rocquencourt et l'ENST de Paris. Le projet Marvel se propose de jeter les bases d'une programmation d'objets logiciels mobiles à grande échelle ainsi que les bases d'une nouvelle infrastructure logicielle répartie mettant en œuvre ce modèle de programmation.

7.3 CTI FT-R&D : Objets réactifs distribués

Participants : Frédéric Boussinot, Jean-Ferdy Susini.

Les résultats de ce contrat (terminé en octobre) ont été les suivants :
- Sémantique du réactif. Dégagement du noyau Junior de programmation réactive en JAVA.
- Implémentations de Junior. Nous avons réalisé plusieurs implémentations de l'approche réactive au dessus de JAVA et dégagé une API qui permet de masquer à l'utilisateur l'implémentation utilisée.
- Évolution des SugarCubes. Deux versions (la v2 et la v3) ont été conçues pendant la CTI.
- Définition du langage Rejo. Rejo (REactive Java Object) est une extension à JAVA qui génère des objets qui peuvent être considérés comme des Objet Réactifs.
- FairThreads. Avec les Fair Threads, on retrouve en fait les caractéristiques fondamentales de la programmation réactive dans un contexte de threads.

7.4 CTI FT-R&D : Objets Migrants

Participants : Raul Acosta, Roberto Amadio, Gérard Boudol, Frédéric Boussinot, Cédric Lhoussaine, Pascal Zimmer.

Les objectifs de ce contrat, qui a débuté en janvier 2000 pour une durée de 3 ans, sont, d'une part, de développer des techniques de modélisation, de spécification et de vérification en ce qui concerne les objets migrants, et d'autre part d'expérimenter la programmation et la validation d'agents mobiles, en se fondant sur le travail déjà accompli en ce qui concerne les objets et Scripts Réactifs. Nous portons un effort particulier sur les aspects de typage, aussi bien en ce qui concerne le modèle objet lui-même que pour ce qui touche aux aspects de migration.

8 Actions régionales, nationales et internationales

8.1 Actions nationales

8.1.1 Actions Concertées Incitatives Cryptologie VERNAM et AZURCRYPT

Participants : Roberto Amadio, Vincent Vanackere.

Nous coordonnons l'Action Concertée Incitative Cryptologie VERNAM sur la Vérification automatique de protocoles cryptographiques. L'action comprend le projet PROTHÉO de l'INRIA-Lorraine et l'ENS-Cachan. Nous participons à l'ACI AZURCRYPT, avec l'IML de Marseille et l'Université de Toulon.

8.2 Actions européennes

8.2.1 Coopération Franco-Portugaise

Participants : Gérard Boudol, Ilaria Castellani, Cédric Lhoussaine, Pascal Zimmer.

Dans le cadre du programme de coopération luso-française de l'ambassade de France et du ICCTI, nous avons un projet de recherche avec l'université de Lisbonne portant sur la sémantique des objets concurrents.

8.3 Actions internationales

8.3.1 NSF/INRIA

Participant : Davide Sangiorgi.

Le thème de cette collaboration avec l'université de Pennsylvanie, intitulée «Static Types for Reasoning about Concurrent Systems», porte sur les modèles de processus mobiles, le typage dans ces formalismes ainsi que leur pouvoir expressif dans la représentation comportementale des objets concurrents.

8.4 Visites, et invitations de chercheurs

Nous avons reçu la visite, pour des périodes dont la durée variait de une journée à deux semaines, des personnes suivantes : Martin Odersky, professeur à l'Ecole Polytechnique Fédérale de Lausanne, un groupe de chercheurs de la société GEMPLUS, autour de Laurent Fournerie, Andrei Sabelfeld, de l'université de Chalmers, Witold Charatonik du MPI de Saarbrück, Petr Jancar de l'université d'Ostrava, Matthew Hennessy et Massimo Merro, de l'université de Sussex, Riccardo Pucella, de l'université de Cornell, Vasco Vasconcelos, Antonio Ravara et Ana Matos, de l'université de Lisbonne, Kohei Honda et Nobuko Yoshida, de Queen Mary College et l'université de Leicester, François Pottier et Vincent Simonet, de l'INRIA Rocquencourt, Peter Sewell, de l'université de Cambridge.

9 Diffusion de résultats

9.1 Animation de la communauté scientifique

• Roberto Amadio est responsable de l'équipe MOVE (Modélisation et Vérification) du Laboratoire d'Informatique de Marseille (CNRS, FRE2246). Il était conférencier invité au Workshop Logical Aspects of Cryptographic Protocols Verification. Il est co-organisateur et membre du comité scientifique du *Programme Logique et Interaction* (CIRM, Marseille). Il est membre du comité de programme de ICALP (track B), 2002 et CONCUR, 2002. Il est Président de la Commission de Spécialistes de l'Université de Provence, Membre titulaire de la Commission de Spécialistes de l'Université d'Aix-Marseille II, expert auprès du ministère pour les Primes d'Encadrement Doctoral et de Recherche et membre du Jury du prix de thèse SPECIF 2001. Il a été *opposant* de la thèse de J.-L. Vivas (U. Stockholm), membre du jury d'habilitation de B. Benhamou (U. de Provence), et co-responsable du séminaire d'informatique du Laboratoire d'Informatique de Marseille.
• Gérard Boudol n'a pu se rendre à la conférence ESOP, où son exposé a été présenté par Ilaria Castellani. Il a donné des exposés à Marseille, Lisbonne et l'INRIA Rocquencourt sur [19], et à Paris (laboratoire PPS) sur [18]. Il a participé à la conférence ICALP. Il est rapporteur de l'habilitation à diriger les recherches de Guiseppe Castagna.
• Frédéric Boussinot a été rapporteur de la thèse de Yann Remond (Université Grenoble 1) et a participé au jury des thèses de Jean-Ferdy Susini (dont il était le directeur de thèse) et Nadine Richard (ENST).
• Ilaria Castellani a donné des conférences sur [11, 18] à Marseille, Universités de Sussex, Lisbonne, Oxford, à la conférence ICALP et à l'ENS Cachan. Elle a participé aux conférences ETAPS, ICALP, CONCUR et EXPRESS. Elle était membre du comité de programme de cette dernière.
• Davide Sangiorgi n'a pu se rendre à la conférence CONCUR, où son exposé a été présenté par Ilaria Castellani. Il supervisait la thèse de Christine Röckl, soutenue en février à Munich. Il a organisé le workshop «Proofs for Mobility» satellite de la conférence ETAPS, à laquelle il a participé. Il a organisé une session spéciale de la réunion IFIP WG 2.2 sur «Mobility and Global Computing». Il a donné une conférence sur les calculs de processus à l'école BASICS à Pekin.
• Pascal Zimmer a donné des exposés sur l'inférence de types pour la récursion à l'université de Lisbonne et à l'INRIA Rocquencourt.

9.2 Enseignements

Raul Acosta-Bermejo, Frédéric Boussinot et Jean-Ferdy Susini (avec la participation de Laurent Hazard) ont donné un cours intitulé «Objets réactifs en JAVA». Ce cours regroupait des étudiants des DEA RSD et Informatique de l'UNSA (Université de Nice-Sophia Antipolis) et des étudiants de 3ème année de l'ESSI. Frédéric Boussinot et Jean-Ferdy Susini ont encadré les stages de DEA de Julien Demaria et Christian Brunette.

Roberto Amadio est responsable de la Licence d'Informatique de l'Université de Provence, responsable du cours «Typage et Vérification» du DEA d'Informatique de Marseille et représentant de l'UFR auprès de l'Institut des Applications Avancées de l'Internet. Il a encadré le

stage de DEA de Charles Meyssonnier et il co-encadre actuellement les thèses de Cédric Lhoussaine, Vincent Vanackère et Charles Meyssonnier. Cédric Lhoussaine est ATER à l'Université de Provence et il intervient dans le DESS Génie Informatique et Statistique.

Gérard Boudol, Ilaria Castellani et Davide Sangiorgi interviennent dans le DEA MDFI à marseille, dans l'option «Calculs de Processus Distribués et Mobiles». D. Sangiorgi a encadré les stages de DEA d'Etienne Lozes et David Teller, tous deux du DEA de l'ENS Lyon.

Pascal Zimmer enseigne en DEUG à l'université de Nice, ou il assure des TD et TP en «Informatique et Programmation», «Algorithmique et Programmation» et «Informatique Théorique».

10 Bibliographie

Ouvrages et articles de référence de l'équipe

[1] R. Amadio, P.-L. Curien, *Domains and Lambda-Calculi*, Cambridge University Press, 1998.

[2] G. Berry, G. Boudol, « The chemical abstract machine », *Theoretical Computer Science 96*, 1992.

[3] G. Boudol, « Asynchrony and the π-calculus », *rapport de recherche n°1702*, INRIA, May 1992, `http://www.inria.fr/rrrt/rr-1702.html`.

[4] F. Boussinot, *La programmation réactive*, Masson, 1996.

[5] F. Boussinot, *Objets réactifs en Java*, Collection Scientifique et Technique des Telecommunications, PPUR, 2000.

[6] I. Castellani, « Process Algebras with Localities », *in : Handbook of Process Algebra*, J. Bergstra, A. Ponse, et S. Smolka (éditeurs), North-Holland, Amsterdam, 2001, p. 945–1045.

[7] D. Sangiorgi, D. Walker, *The π-Calculus : a Theory of Mobile Processes*, Cambridge University Press, 2001.

Thèses et habilitations à diriger des recherches

[8] J.-F. Susini, *L'approche réactive au dessus de* Java *: implémentation et sémantique des SugarCubes et de Junior*, thèse de doctorat, Ecole des Mines de Paris, octobre 2001.

Articles et chapitres de livre

[9] R. Acosta, « Reactive operating system/Reactive Java Objects », *RERIR*, 2001.

Communications à des congrès, colloques, etc.

[10] R. Amadio, C. Meyssonnier, « On the decidability of fragments of the asynchronous π-calculus », *in : Proc. EXPRESS01, Electronic Notes in Theoretical Computer Science, 52.1*, 2001. Also appeared as RR-INRIA 4241, `http://www.inria.fr/rrrt/rr-4241`.

[11] G. Boudol, I. Castellani, « Noninterference for Concurrent Programs », *in : ICALP'01, Lecture Notes in Computer Science*, 2076, p. 382–395, 2001.

[12] G. Boudol, « The recursive record semantics of objects revisited », *in : ESOP'01, LNCS, 2028*, 2001.

[13] F. Boussinot, J.-F. Susini, F. Dang Tran, L. Hazard, « A reactive behavior framework for dynamic virtual worlds », *in : Web3D*, 2001.

[14] D. Sangiorgi, A. Valente, « A distributed abstract machine for Safe Ambients », *in : ICALP'01*, *LNCS*, *2076*, 2001.

[15] D. Sangiorgi, D. Walker, « Some results on barbed equivalences in π-calculus », *in : CONCUR'01*, *LNCS*, *2154*, 2001.

[16] D. Sangiorgi, « Extensionality and intensionality of the Ambient Logics », *in : POPL'01*, 2001.

Rapports de recherche et publications internes

[17] R. Amadio, D. Lugiez, V. Vanackere, « On the symbolic reduction of processes with cryptographic functions », *Report 4147*, INRIA, 2001, soumis pour publication, `http://www.inria.fr/rrrt/rr-4147`.

[18] G. Boudol, I. Castellani, « Noninterference for Concurrent Programs and Thread Systems », *Report 4254*, INRIA, 2001, accepté pour publication dans TCS., `http://www.inria.fr/rrrt/rr-4254`.

[19] G. Boudol, « The recursive record semantics of objects revisited », *Report 4199*, INRIA, 2001, soumis pour publication., `http://www.inria.fr/rrrt/rr-4199`.

[20] F. Boussinot, « Java fair threads », *Report 4139*, INRIA, 2001, `http://www.inria.fr/rrrt/rr-4139`.

[21] C. Lhoussaine, « Type Inference for Dπ_1^r », *Report to appear*, INRIA, 2001.

[22] P. Zimmer, « On the Expressiveness of Pure Safe Ambients », *Report 4350*, INRIA, 2001, accepté pour publication dans MSCS., `http://www.inria.fr/rrrt/rr-4350`.

Divers

[23] R. Acosta, « Programming in Rejo », à paraître dans RERIR, 2001.

[24] C. Brunette, « Etude de la cohérence dans les jeux en réseaux distribués », Rapport de DEA, Université de Nice-Sophia Antipolis, 2001.

[25] J. Demaria, « Programmation réactive fonctionnelle avec Senior », Rapport de DEA, Université de Nice-Sophia Antipolis, 2001.