IMSA '92 Workshop on reflection and meta-level architectures

Tokyo 4 - 7 novembre 1992

Rubrique réalisée en coopération avec le CNRS-JAPON, sous la responsabilité scientifique de Jean-Pierre Briot, Chargé de Mission CNRS en Sciences pour l'Ingénieur auprès du CNRS-JAPON

La lettre Japon IA publiée par le Service pour la Science et la Technologie de l'Ambassade de France au Japon ouvre ses colonnes à partir de ce numéro aux chercheurs du CNRS en poste au Japon. Ces chercheurs, membres d'un laboratoire du CNRS en France, travaillent au Japon dans le cadre de projets communs à leur laboratoire d'origine et leur laboratoire d'accueil au Japon. Leur présence au sein d'une équipe de recherche, ainsi que les contacts qu'ils nouent sont riches d'informations et de propositions qui contribueront sans aucun doute à mieux faire saisir les enjeux, les projets scientifiques japonais mais aussi la "manière" dont se pratique la science au quotidien dans ce pays. Gageons que cette collaboration avec Japon IA sera enrichissante pour tous.

Jean-François Sabouret
Directeur du CNRS-Japon

Du 4 au 7 Novembre 1992 a eu lieu à Tokyo un workshop (groupe de travail) international sur une des directions de recherche récentes et novatrices en matière d'informatique. Il s'agit de la capacité d'introspection appelée "réflexion" (en anglais "reflection"), permettant ainsi à un système de s'adapter et s'optimiser. Ce groupe de travail a réuni la plupart des chercheurs les plus en pointe dans ce domaine, avec notamment une importante contribution japonaise. Dans ce rapport, nous signalerons plus particulièrement les contributions japonaises présentées à ce groupe de travail ou lors de rencontres précédentes.

Organisation

Ce groupe de travail a été organisé par Kokichi Futatsugi de l'Electrotechnical Laboratory (ETL, Tsukuba). Le comité de programme était co-dirigé par deux chercheurs de pointe du domaine : Akinori Yonezawa, Professeur à l'Université de Tokyo, et Brian Smith, chercheur au Xerox Parc (Palo Alto Research Center). Brian Smith est l'inventeur du concept de "réflexion" (avec le système 3-Lisp au début des années 80).
Cette rencontre a été financée conjointement par le Research Institute of Software Engineering (RISE) et par l'Information Technology Promotion Agency (IPA). Ce groupe de travail fait partie d'un programme de recherche (programme JISEDAI) appelé "New Models for Software Architecture", financé par le MITI.

Méta-programmation et réflexion

Vers une meilleure adaptation

Nos besoins informatiques sont de plus en plus importants. Au fur et à mesure que les outils informatiques progressent en complexité et en efficacité nous les appliquons à des problèmes plus importants, plus complexes et surtout plus évolutifs. Les systèmes informatiques (par systèmes informatiques, nous incluons interprètes et compilateurs de langages de programmation, applications...) doivent pouvoir s'adapter non seulement à des problèmes particuliers de manière statique (avant de démarrer les calculs) mais également de manière dynamique, c'est-à-dire au cours de l'évolution des calculs. Les informaticiens sont donc à la recherche de méthodologies permettant à la fois de configurer des systèmes de manière modulaire et également d'assurer leur adaptation aux ressources disponibles au cours du calcul, en vue d'une efficacité optimale.

Méta-programmation

Pour changer le comportement d'un système informatique et pouvoir l'optimiser, les informaticiens savent descendre au niveau de l'implémentation (au dessous du niveau de spécification/programmation fourni par le système). Le problème est que le code devient alors peu lisible et vérifiable, et surtout très difficile à faire évoluer ou réutiliser.
L'idée est donc de contrôler le comportement du système non pas à un très bas niveau d'expression, mais à un niveau au moins équivalent à celui du système lui-même. On parle alors de "méta-programmation", c'est-à-dire le moyen de décrire et contrôler le déroulement d'un programme. Par exemple dans certains systèmes experts à règles de production, des "méta-règles" permettent de contrôler le déclenchement des règles (un des problèmes étant l'ordonnancement dans le cas de plusieurs règles candidates). Le "méta-niveau" représente et décrit l'exécution du

"niveau de base", c'est-à-dire le déroulement du programme. Il est possible que le méta-niveau ait lui-même un méta-niveau et ainsi de suite. En l'occurrence on considère que trois niveaux sont suffisants pour la plupart des problèmes.
De telles architectures à méta-niveaux sont devenues relativement répandues en intelligence artificielle. Elles permettent d'exprimer la capacité d'un système à raisonner sur son propre comportement (par exemple pour détecter des blocages lors de raisonnements et pouvoir les résoudre).

Réflexion

Une question laissée ouverte est la suivante : "Dans quel langage est spécifié le méta-niveau?". Une réponse naturelle est de choisir le même langage que pour écrire les programmes dans le système lui-même. On parle alors de système "réflexif". On peut donc décrire et contrôler le fonctionnement du système à partir du système lui-même. Pour ce faire le système possède une représentation de son propre fonctionnement. Le système et sa représentation sont dits "causalement connectés", c'est-à-dire que toute modification dans l'état du système se trouve reflétée dans sa représentation et vice-versa. La séparation explicite entre le système et sa représentation (décrivant son implémentation et son fonctionnement) permet d'éviter de faire la confusion entre le système en tant qu'exécuteur de programme et le système en tant que contrôle de cette exécution.

Réification et réflexion

L'opération dite de réification permet de passer du système à sa représentation. On peut alors manipuler les composantes de l'état du ou des calculs en cours représentées sous forme d'entités. L'opération duale dite de "réflexion" permet de réimplémenter la représentation ainsi modifiée dans le système lui-même. L'opération de réification passe du niveau de base au niveau méta, et l'opération de réflexion permet d'y redescendre.

Exemple d'application

Pour concrétiser un peu plus notre propos qui est resté jusqu'ici relativement abstrait, nous allons maintenant présenter brièvement deux exemples :
• Implémentation d'un système de coroutines (sous-programmes se passant la main) :
L'idée est de stopper le calcul en cours, de capturer l'état de calcul (notamment la suite du calcul, appelée "continuation"), de démarrer l'exécution d'un autre sous-programme ainsi suspendu. Le langage de programmation Scheme (un dialecte de Lisp, un des langages clés de l'informatique avancée) rend immédiate cette extention de par la manipulation explicite de la continuation du calcul (réifiée sous forme de fonction)
• Spécialisation /optimisation d'un système de multi-fenêtrage :
L'idée est cette fois d'implanter un tableur à partir d'un système de multi-fenêtrage. Cette approche correspond à une intuition naturelle (une cellule de tableur est un cas dégénéré de fenêtre) mais n'est pas utilisable en pratique par manque d'efficacité (notamment par une occupation mémoire superflue). La réflexion permet de particulariser le système de multi-fenêtrage en réduisant sa complexité et ses fonctionnalités (système Silica de Ramana Rao, Xerox Parc).
Le premier exemple est un exemple de réflexion dite "procédurale" (on intervient sur la représentation des entités du programme). Le deuxième exemple est un exemple de réflexion dite "structurale" , par opposition à la réflexion "procédurale" (on intervient sur le déroulement du programme). Ces deux aspects sont complémentaires. Dans les deux cas la réflexion permet d'"ouvrir" et de rendre modulaire le contrôle du comportement de programmes, en particulier leur optimisation.

Concurrence

Jusqu'à maintenant nous sommes restés implicitement dans un contexte séquentiel. La réification et la réflexion respectivement suspend le déroulement du programme puis reprend son exécution.
Le passage à un univers concurrent (c'est-à-dire doté de parallélisme potentiel) s'avère naturel. Chaque processus de calcul se trouve donc doté d'un niveau de description et de contrôle. Le parallélisme potentiel entre un processus et son contrôle permettent par exemple une intervention en cas de blocage du processus. Mais il faut veiller à tenir compte des inconsistances éventuelles entre un processus et sa représentation en cas de concurrence entre eux. Voir à ce sujet les propositions méthodologiques de Tomoyuki Tanaka du Centre scientifique IBM (présentées il y a deux ans). Il distingue entre réification "gelante" (qui suspend le processus en cours tant que l'on opère au méta-niveau) ou "non-gelante" (concurrence libre, dans le cas où une garantie de consistance n'est pas nécessaire : dans le cas de production de statistiques par exemple).
L'étude de la réflexion dans des langages de programmation concurrente a été particulièrement développée au Japon, notamment à l'Université de Tokyo par Akinori Yonezawa et son équipe

autour du projet ABCL. Signalons également dans le domaine des langages de programmation concurrente logique les contributions de Jiro Tanaka sur le développement de Meta-GHC (c'est-à-dire un Guarded Horn Clauses réflexif) dans le cadre du projet ICOT, et de Hiroyasu Sugano (Fujitsu).

Localité de la représentation

Deux questions sont restées ouvertes jusqu'à maintenant. Le premier point concerne le choix de représentation. On peut représenter l'état de l'ensemble du système (c'est ainsi le cas pour le langage séquentiel Scheme). On peut également décomposer/répartir les représentations associées aux différents éléments ou perspectives composant le système. Ainsi dans un langage de programmation dit "à objets", le programme est éclaté dans un ensemble d'entités communicantes. On peut donc associer une représentation (appelée "méta-objet") à chaque objet du système.
On peut augmenter cette décomposition en associant à chaque objet plusieurs méta-objets, chacun concernant un point de vue (par exemple les ressources, les statistiques, la répartition physique...). Cette approche permet de décomposer les différentes perspectives de l'exécution d'un objet dans son environnement. Cette direction est représentée par le langage AL-1/D de Hideaki Okamura, de l'équipe de Mario Tokoro à l'Université Keio, Tokyo, et Yutaka Ishikawa de ETL.
A l'inverse si l'on veut décrire les relations des objets à l'intérieur d'un environnement ou une ressource commune (par exemple pour exprimer le séquencement d'un ensemble d'objets/processus sur un processeur), il nous faut introduire la notion de "réflexion de groupe". Dans ce cas un tel méta-objet décrit la cohésion d'un ensemble d'objets. Cette approche a été proposée par Takuo Watanabe de l'Université nouvellement créée: Japan Advanced Institute of Science and Technology (JAIST, Hokuriku).
On le voit : comme dans tout système de représentation, tout est question de point de vue! Il faut pouvoir combiner ces différents points de vue et en changer à volonté. Une architecture "hybride", intégrant réflexion individuelle et de groupe a d'ailleurs été proposée en ce sens par l'équipe de Akinori Yonezawa à l'Université de Tokyo. Le langage de programmation résultant, descendant du langage de programmation concurrente à objets ABCL, se nomme RbCl (Yuuji Ichisugi et Satoshi Matsuoka).

Applications

Des exemples d'application abordés dans le groupe de travail sont :
•l'extension d'un langage de programmation par de nouveaux mécanismes (par exemple un mode pas à pas, un mécanisme d'héritage, ou encore un mécanisme de traitement de contraintes temps-réel),
•l'implémentation de systèmes d'exploitation répartis hétérogènes (ainsi le système réparti Apertos, développé par Yasuhiko Yokote au laboratoire de Sony CSL dirigé par Mario Tokoro),
•mieux structurer et améliorer les systèmes d'IA (notamment la capacité d'apprentissage par reconnaissance d'analogies avec des situations déjà rencontrées (et réifiées), ainsi que la capacité d'explication),
•rendre plus flexibles et adaptatifs les modules logiciels (c'est donc en cela que le thème de la réflexion est au coeur du projet "New Models for Software Architecture"). L'idée est que les modules logiciels aient une représentation d'eux-mêmes et puissent acquérir des représentations (modèles) des autres modules avec lesquels ils s'interfaceront pour pouvoir, sans l'intervention de programmeurs, négocier leur interfaçage. On touche donc le domaine des systèmes coopératifs (IA distribuée, multi-agents). Remarquons dans ce cadre la contribution de Hideyuki Nakashima (ETL) sur une "réflexion contextuelle", où chaque agent peut changer son comportement en modifiant ses contextes décrivant (réifiant) son environnement.

Conclusion

Problèmes ouverts et directions futures

Un des problèmes laissés encore relativement ouvert est celui de la protection. L'utilisation de la réflexion, outil surpuissant par essence, peut aboutir jusqu'à "casser" un système si elle n'est pas entourée d'une certaine discipline. Il faut sans doute chercher à s'inspirer de méthodologies éprouvées du génie logiciel.
Un autre problème est que le terme reste encore relativement vaguement défini (un nouveau mot attrape-mouche, après les objets, disent certains!). Mais nous pensons que, comme pour les objets,

la force du concept réside justement dans sa généricité et sa capacité à décrire différents domaines. L'exploration du concept de réflexion est encore en phase d'expansion, comme le souilignait Peter Deutsch (Sun Labs) au cours de la conclusion du groupe de travail. De nombreux domaines ont encore été peu explorés à l'aide de ce filtre. Cependant des techniques relevant de la méta-programmation et de la réflexion sont déjà appliquées dans certaines spécialités (nous avons déjà mentionné l'IA, citons également l'optimisation de requêtes de bases de données). Comme pour Monsieur Jourdain, ces chercheurs découvrent ainsi "sur le tard" qu'ils programmaient déjà réflexif sans le savoir!
Pour formaliser les concepts, des tentatives de description formelle commencent à être proposées, notamment une sémantique dénotationnelle en cours (Shin Nakajima, C&C Systems Research Labs., NEC), et une spécification algébrique (Masahito Kurihara, Université d'Hokkaido).

Bilan

Nous avons pu constater une nouvelle fois à l'occasion de ce groupe de travail que la recherche japonaise est bien représentée dans ce domaine. Deux groupes sont prépondérants : l'équipe de Akinori Yonezawa à l'Université de Tokyo, et l'équipe de Mario Tokoro, répartie entre l'Université Keio et le laboratoire Sony CSL. Ceci n'est pas pour nous surprendre car il se trouve que ces deux équipes sont également prépondérantes dans le domaine de la programmation concurrente par objets.
Enfin terminons en soulignant que le concept de réflexion de groupe versus réflexion individuelle a été proposé dans une équipe japonaise (par Takuo Watanabe). Peut-on alors parler de (méta-)"réflexion" à propos du comportement social de ce pays?! ❑

Jean-Pierre Briot
Department of Information Science, Faculty of Science,
The University of Tokyo
Hongo, Bunkyo-ku, Tokyo 113
tel: (03) 38 12 21 11 ext. 4120
ligne directe: (03) 58 00 69 13
fax: (03) 56 89 43 65
e-mail: briot@is.s.u-tokyo.ac.jp

CNRS-Japon
Hanzomon MK Bldg. 6F, 1-8-1, Kojimachi, Chiyoda-ku, Tokyo 102 JAPON
tel : +81-3-3288-7575/6/8 ; Fax : +81-3-3288-7577