INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Projet CONTRAINTES

Programmation par Contraintes

Rocquencourt

THÈME 2A

Table des matières

N.B. Contraintes est conçu pour devenir un projet commun avec l'Université de Paris 6.

1 Composition de l'équipe

Responsable scientifique
François Fages [DR, INRIA]
Responsable permanent
Pierre Deransart [DR, INRIA]
Assistante de projet (en commun avec Atoll)
Josy Baron [AJT, INRIA]
Personnel INRIA
Jean-Claude Sogno [DR]
Conseiller scientifique
Philippe Codognet [Professeur, Université de Paris 6]
Collaborateurs extérieurs
Frédéric Benhamou [Professeur, Université de Nantes]
Daniel Diaz [Maître de conférence, Université de Paris 1]
Gérard Ferrand [Professeur, Université d'Orléans]
Doctorants
Emmanuel Coquery [Bourse MESR, INRIA]
Sorin Craciunescu [AMX, INRIA]
Yoann Fabre [Bourse MESR, Paris 6]
Luc Hernandez [Bourse MESR, Paris 6]
Ludovic Langevine [Bourse INRIA région, à partir du 1/10/2001]
Nadine Richard [ATER Paris 6, jusqu'au 31/08/2001]
Sylvain Soliman [Ingénieur DGA, INRIA jusqu'au 30/04/2001]
Stagiaires
Rémy Haemmerlé [du 9/4/2001 au 21/9/2001]
Ludovic Langevine [du 1/7/2001 au 30/9/2001]
Jean-Michel Leconte [du 9/8/2001 au 30/8/2001]
Alexis Saurin [du 5/6/2001 au 21/9/2001]
Rachid Zoumman [du 1/4/2001 au 21/9/2001]

2 Présentation et objectifs généraux

Le projet Contraintes s'intéresse à la programmation par contraintes, de différents points de vue : conception de nouveaux langages et de leurs environnements de programmation, fondements sémantiques, conception de solveurs de contraintes, et exploration de nouvelles applications.

Le domaine de la programmation par contraintes est né il y a une quinzaine d'années d'un rapprochement de la programmation logique, de la programmation linéaire venant de la Recherche Opérationnelle, et des techniques de propagation de contraintes issues de l'Intelligence Artificielle. Dans son principe, la programmation par contraintes consiste simplement à

programmer avec des variables mathématiques et des relations entre ces variables, soit des relations primitives, appelées contraintes, soit des relations définies par programme. Le modèle de machine classique se trouve ainsi remplacé par un modèle de machine abstraite de contraintes, dans lequel la mémoire n'est plus une table de valeurs mais un ensemble de contraintes. L'opération d'écriture en mémoire est devenue une opération d'ajout de contrainte, et l'opération de lecture est un test d'implication de contrainte.

On peut dire qu'aujourd'hui les techniques de programmation par contraintes participent à un véritable renouveau de la Recherche Opérationnelle pour la résolution des problèmes combinatoires en milieu industriel, et que ce succès illustre principalement l'apport des recherches sur les *langages déclaratifs* pour combiner efficacement des techniques de résolution hétérogènes : numériques, symboliques, déductives, heuristiques. L'utilisation de langages de haut niveau pour la modélisation à la fois du problème à résoudre et des stratégies de résolution, entraîne une diminution importante des coûts de développement et de maintenance du logiciel. Cette contribution de génie logiciel permet aussi d'attaquer des problèmes nouveaux, NP-difficiles, dans leur globalité, ce qui était difficilement envisageable sans des outils adaptés pour les programmer.

Le traitement de nouvelles applications motive la recherche de nouvelles extensions des outils existants, dans le but de traiter de nouveaux domaines de contraintes, de concevoir de nouveaux solveurs ou de nouvelles combinaisons de solveurs, ainsi que de nouvelles procédures de recherche. Les outils de programmation par contraintes pourraient aussi être plus faciles d'utilisation, et toucher un plus grand nombre d'utilisateurs. La mise au point des programmes, la détection des erreurs, l'interrogation de ce qui se passe à l'exécution, ne sont pas des problèmes bien résolus aujourd'hui.

Nos travaux portent donc à la fois sur les langages de programmation par contraintes, leurs fondements sémantiques, leurs techniques d'implémentations, leurs environnements de mise au point ; et sur les techniques de résolution de contraintes, exactes ou approchées, règles de simplification, propagation de contraintes, recherche locale.

L'étude des langages concurrents avec contraintes (CC) se trouve au centre du projet. Elle fournit un cadre conceptuel commun pour étudier plusieurs aspects pourtant bien distincts de la programmation par contraintes, comme les techniques de résolution de contraintes, les langages de modélisation multiple concurrente, les implantations distribuées, les applications réactives.

Le projet Contraintes s'intéresse principalement à deux domaines applicatifs : la résolution de problèmes combinatoires, domaine d'excellence de la programmation par contraintes aujourd'hui ; et la réalité virtuelle, domaine plus prospectif où les contraintes apparaissent comme un paradigme de programmation déclarative pour la création de mondes virtuels et la spécification des comportements d'agents évoluant dans un environnement virtuel 3D.

3 Fondements scientifiques

3.1 Langages concurrents avec contraintes

Mots clés : programmation logique, programmation par contraintes, solveurs de

contraintes, programmation concurrente, communication, synchronisation, distribution.

La classe des langages concurrents avec contraintes, introduite par V. Saraswat au début des années 90, est une abstraction représentative des systèmes de programmation par contraintes, qui se prête bien à l'étude de leurs propriétés fondamentales, tant en termes d'expressivité, de complexité, de modèles d'exécution ou de coopération, que de méthodes de vérification.

Cette classe de langages, notée CC($\mathcal{X}$), est paramétrée par le domaine du discours $\mathcal{X}$ donné avec son langage de contraintes (arithmétique entière ou réelle, linéaire ou non, domaines finis, contraintes symboliques, etc.). Elle généralise la classe CLP($\mathcal{X}$) des programmes logiques avec contraintes par l'introduction d'une primitive de synchronisation basée sur l'implication de contraintes, et constitue de ce fait un paradigme de programmation concurrente. Schématiquement les agents CC partagent des variables (qui jouent le rôle de canaux de communication dynamiques comme en π-calcul asynchrone) et une mémoire de contraintes portant sur ces variables, dans laquelle chaque agent peut écrire, par l'opération d'ajout de contrainte, et lire, par l'opération de test d'implication de contraintes.

Un des succès notoires du cadre CC a été la reconstruction simple et élégante de solveurs de contraintes sur les domaines finis, ainsi que la coopération de modélisations multiples pour la résolution de problèmes combinatoires. L'utilisation du cadre CC pour la programmation de systèmes réactifs nécessite quant à elle de rompre avec l'hypothèse d'évolution monotone de la mémoire des contraintes, et d'autoriser le retrait de contraintes pour la prise en compte de l'évolution temporelle du problème.

Ces préoccupations motivent nos travaux sur de nouvelles extensions des langages CC.

3.1.1 Langages CC et logique linéaire

Mots clés : logique linéaire, logique non-commutative, espaces de phases, model checking, calcul des séquents, démonstration automatique.

Les théorèmes de complétude qui existent entre l'exécution des programmes CLP et leur traduction en logique classique, et qui sont à la base de méthodes simples et puissantes de raisonnement sur les programmes, ne résistent pas à l'opération de synchronisation des programmes CC. La recherche d'une sémantique logique des langages CC, dans le paradigme général de la programmation logique — programme=formule, exécution=recherche de preuve — conduit à une traduction des programmes CC dans la logique linéaire de Jean-Yves Girard. Cette traduction permet de caractériser les ensembles de contraintes accessibles et les succès [9]. La traduction dans la logique non-commutative de Ruet-Abrusci permet en outre de caractériser les suspensions.

Ces résultats ouvrent des perspectives nouvelles pour aborder plusieurs problèmes importants de la programmation par contraintes aujourd'hui :

- valider les programmes concurrents avec contraintes ;
- combiner propagation de contraintes et changement d'état ;
- faire coopérer des méthodes de propagation locale avec des solveurs de contraintes globales.

L'approche utilisée dans les deux derniers cas repose sur une extension naturelle des langages CC, nommée LCC, qui consiste simplement à considérer des systèmes de contraintes

fondés sur la logique linéaire au lieu de la logique classique. Cette généralisation des systèmes de contraintes permet de rendre compte des changements d'états par le biais des consommations de ressources dans la mémoire des contraintes, lors de l'opération de synchronisation modélisée par l'implication linéaire.

3.2 Solveurs de contraintes

Mots clés : langages de contraintes, domaines de contraintes, domaines finis, contraintes linéaires, contraintes non-linéaires, contraintes floues, problèmes surcontraints, coopération, recherche locale, propagation de contraintes.

Les domaines applicatifs que nous considérons utilisent différents systèmes de contraintes, en particulier :

- contraintes sur les domaines finis (nombres entiers naturels bornés) : contraintes primitives d'appartenance à un domaine fini, contraintes numériques, symboliques, contraintes globales, contraintes d'ordre supérieur ;
- contraintes sur les réels : algorithme du Simplexe pour les contraintes linéaires, méthodes d'intervalles pour les contraintes non-linéaires ;
- contraintes valuées dans des demi-anneaux : contraintes floues, hiérarchiques, traitement des problèmes surcontraints.

Cette diversité des algorithmes de résolution pose le problème fondamental de la coopération des solveurs, que ce soit au sein d'un même domaine de contraintes pour accélérer la résolution ou traiter des contraintes de types différents, ou entre différents domaines pour traiter les aspects hétérogènes d'un problème, ou bien des relaxations du problème.

Le projet travaille sur les algorithmes de résolution de contraintes et leurs méthodes de coopération. En particulier nous cherchons à faire coopérer les méthodes de recherche locale avec les méthodes de propagation de contraintes, par exemple pour la définition des voisinages.

3.3 Environnements de programmation

Mots clés : mise au point, débogage, trace, visualisation, typage, preuve de programmes, vérification.

Le traitement de plusieurs centaines ou milliers de contraintes hétérogènes sur autant de variables est un processus qu'il n'est pas vraiment possible d'appréhender sans des outils de visualisation des différents aspects de l'exécution, des environnements de mise au point des programmes, et des méthodes d'analyse et de vérification des programmes.

Le projet Esprit DiSCiPl a montré d'une part l'apport des recherches conduites sur le diagnostic déclaratif d'erreurs fondé sur la sémantique logique des programmes, pour la conception de systèmes interactifs de débogage des programmes, et a développé d'autre part un ensemble d'outils puissants de visualisation de l'exécution des programmes CLP sous ses différents aspects : effets de la propagation des contraintes, forme de l'espace de recherche (avec détection des isomorphismes de sous-arbres, révélateurs de symétries non exploitées), impact des heuristiques, etc.

Les recherches se poursuivent, dans le cadre du projet RNTL OADymPPac, sur les méthodes de visualisation des contraintes et de l'espace de recherche, sur les méthodes de débogage fondées sur la sémantique des programmes, sur les systèmes de typage statique et dynamique, et plus généralement sur les méthodes de validation des programmes concurrents avec contraintes.

4 Domaines d'applications

4.1 Optimisation combinatoire

Mots clés : télécommunication, ingénierie, transport, ordonnancement, allocation de ressources, placement, planification.

Les problèmes d'optimisation combinatoire rencontrés dans le monde industriel sont de plus en plus fréquents, et leur impact économique va croissant. Ces problèmes concernent :
- l'allocation de ressources ;
- le placement ;
- l'ordonnancement de tâches ;
- la parallélisation ;
- la planification de personnel, rotation d'équipages ;
- le transport ;
- le pilotage d'équipements multimode ;
- etc.

Les travaux réalisés en Recherche Opérationnelle depuis 40 ans ont abouti à une panoplie impressionnante de techniques de résolution dans des domaines variés : programmation linéaire, programmation linéaire en nombres entiers, optimisation par séparation-évaluation, relaxation Lagrangienne, recherche locale, heuristique, algorithmes sur les graphes.

L'apport de la programmation par contraintes se situe d'une part dans les techniques de cohérence locale de contraintes qui s'appliquent à une grande variété de contraintes numériques ou symboliques, et d'autre part dans la conception de langages de programmation déclaratifs qui permettent de modéliser le problème combinatoire à l'aide de relations, combiner différentes modélisations et différentes techniques de résolution, spécifier les heuristiques, et définir les procédures d'exploration de l'espace de recherche.

L'apport des langages déclaratifs pour ces différents aspects est un apport de génie logiciel essentiel, car il permet d'expérimenter des combinaisons d'algorithmes qu'il serait difficile de programmer sans langages de suffisamment haut niveau d'abstraction. Cela explique l'obtention, en ordonnancement par exemple, de résultats meilleurs que ceux obtenus par des approches classiques, mais cela vaudra encore plus à l'avenir pour la coopération des méthodes de résolution globale et de propagation locale, ou la définition des procédures de recherche.

Le projet Contraintes s'appuie dans ce domaine sur sa maîtrise des langages concurrents avec contraintes, des solveurs de contraintes et de leurs implémentations. Les recherches sur LCC fournissent un cadre théorique pour traiter plusieurs aspects de la programmation par contraintes qui se formalisent difficilement dans les cadres existants. Les travaux sur les contraintes floues permettent d'attaquer des applications nouvelles d'optimisation combinatoire comportant des problèmes surcontraints. Enfin nos travaux sur les environnements de

mise au point répondent aux besoins spécifiques de recherches visant à améliorer la productivité et faciliter l'utilisation des outils de programmation par contraintes dans ces domaines applicatifs.

4.2 Réalité virtuelle

Mots clés : multimédia.

L'utilisation d'environnements 3D et de mondes virtuels, que ce soit dans des dispositifs immersifs complexes ou sur un classique écran de station de travai,l se développe pour de nombreuses applications, du magasin virtuel pour le commerce électronique aux jeux vidéos. La conception et l'implantation de mondes virtuels restent cependant des tâches difficiles car il existe peu d'outils de haut niveau permettant une réalisation aisée. En effet, si de nombreux logiciels puissants ont été développés en ce qui concerne la modélisation et le rendu 3D, il existe un manque certain de systèmes capables de gérer non seulement des objets animés mais surtout des agents autonomes basés sur des comportements réactifs particuliers. En outre, l'extension de tels systèmes à des environnements distribués standard (Internet) pose des problèmes sortant du champ de l'informatique graphique, et pour lesquels des techniques logicielles issues des paradigmes de programmation des langages déclaratifs peuvent être utilisées. Ainsi la réalisation d'environnements virtuels distribués peuplés de « créatures » autonomes intelligentes est un domaine de recherche ouvert et à fort impact applicatif. Dans ce but, l'utilisation de langages de programmation de haut niveau et de techniques de résolution de contraintes pour spécifier et implanter des relations complexes entre objets animés et décrire des comportements d'agents autonomes est une direction de recherche prometteuse que nous étudions.

5 Logiciels

5.1 GNU-Prolog

Participant : Daniel Diaz [correspondant].

GNU-Prolog est un compilateur natif pour Prolog intégrant un puissant solveur de contraintes sur les domaines finis. GNU-Prolog accepte donc des programmes Prolog avec contraintes et produit des exécutables (binaires) entièrement autonomes dont la taille reste réduite. GNU-Prolog offre également la possibilité d'exécuter un programme à l'aide de l'interprète pour en faciliter la mise au point. En termes de performances, GNU-Prolog est comparable aux meilleurs systèmes commerciaux. GNU-Prolog est un système complet, robuste, efficace et compatible avec la norme ISO pour Prolog mais offrant également bon nombre d'extensions telles que : variables globales, tableaux, interface avec le système d'exploitation sous-jacent, *sockets*, etc. De plus il intègre un solveur de contraintes très efficace, alliant ainsi la puissance de la programmation par contraintes à l'aspect déclaratif de la programmation logique. GNU-Prolog est donc prêt pour une utilisation dans des applications de grande envergure.

Les développements à l'origine de GNU-Prolog ont débuté en 1996 et ont duré 3 ans. Fin 1998 nous avons entamé une discussion avec l'organisation GNU (www.gnu.org) dont le but est de promouvoir le logiciel libre. L'adoption de notre système par GNU pour être *le* Prolog

de GNU a eu lieu en mars 1999 (www.gnu.org/software/prolog). La première version a été diffusée par internet en avril 1999. Cette année le travail a porté plus particulièrement sur le portage sous Windows de GNU-Prolog. A ce jour plus de 30000 exemplaires ont été téléchargés depuis le site FTP de l'INRIA (nous ne disposons pas de statistiques concernant le site FTP de GNU ni les nombreux miroirs de par le monde). GNU-Prolog est aujourd'hui inclus dans les distributions majeures de Linux telles que : Debian, Mandrake, RedHat, ...

Nous avons également mis en place un projet *SourceForge* pour le développement de GNU-Prolog (voir `http://gprolog.sourceforge.net/`). Le but de ce site est d'accueillir des projets *Open Source* et de permettre à des développeurs du monde entier de collaborer sur un projet. Trois modules sont disponibles via un gestionnaire de versions (CVS) sur ce site : les sources, la documentation et les exemples. Toute personne peut télécharger les dernières versions d'un module (CVS en mode lecture), proposer des modifications ou des *patches*, discuter au travers d'un forum,... Les personnes désireuses de s'impliquer davantage dans le développement font une demande auprès d'un des administrateurs qui peut alors donner les droits d'écriture sur l'archive CVS. Nous espérons que cet effort contribuera à favoriser les développements extérieurs.

5.2 TCLP

Participant : Emmanuel Coquery [correspondant].

TCLP est un programme qui vise à typer les programmes écrits en Prolog et dans leurs extensions de programmation par contraintes [8]. Moyennant une description des types des différents symboles de fonction et optionnellement des prédicats, le système infère le type des variables et des prédicats sans type, et vérifie que les différentes clauses et les différents buts du programme sont bien typés. Ces vérifications permettent de détecter à la compilation des erreurs classiques de programmation, telles que l'inversion d'arguments, ou le mauvais usage de prédicats définis dans d'autres modules.

TCLP est fondé sur le système de types proposé par Fages et Paltrinieri pour les programmes logiques avec contraintes. Ce système combine polymorphisme paramétrique et polymorphisme de sous-typage. Les relations de sous-typage permettent de typer non seulement les diverses coercions entre domaines de contraintes, mais également les prédicats de métaprogrammation en Prolog, à travers l'utilisation de relations de sous-typage très générales entre constructeurs de types d'arités différentes, comme par exemple $list(\alpha) < term$ qui permet de traiter les listes comme des termes.

L'implémentation réalisée ainsi qu'une version de démonstration en ligne sont disponibles à l'URL `http://contraintes.inria.fr/~coquery/tclp`. TCLP utilise actuellement la bibliothèque Wallace, développée par François Pottier dans le projet Cristal, pour la résolution des contraintes de sous-typage. Cette partie est actuellement en cours de réécriture dans les *Constraint Handling Rules* (CHR) au dessus de Prolog. Le système TCLP a été testé sur les prédicats prédéfinis de GNU-Prolog et SICStus Prolog, sur les bibliothèques de SICStus Prolog (plus de 3000 prédicats) et sur une implantation en Prolog de CLP($\mathcal{FD}$) (130 prédicats utilisant fortement la métaprogrammation).

5.3 Simplexe

Participants : Jean-Claude Sogno [correspondant], Rémy Haemmerlé.

Un solveur de contraintes linéaires en nombres réels (simplexe), réalisé en langage C par Jean-Claude Sogno, est disponible. Son principe (méthode classique) permet des estimations d'erreur d'arrondi avec une précision meilleure que celle que permettrait la méthode révisée du Simplexe, ce qui le rend bien adapté aux problèmes de taille « moyenne » (quelques centaines de variables, quelques centaines de contraintes). Ce solveur a été modifié par Jean-Claude Sogno pour en permettre une utilisation incrémentale.

L'intégration de ce solveur de contraintes dans GNU-Prolog a été réalisée par Rémy Haemmerlé, en fournissant une implémentation de CLP($\mathcal{R}$). L'interface réalisée pourra être réutilisée pour connecter GNU-Prolog avec d'autres implémentations de la méthode du Simplexe ou d'autres méthodes.

5.4 CLP($\mathcal{FD}$,$\mathcal{S}$)

Participant : Philippe Codognet [correspondant].

Ce logiciel, réalisé par Yan Georget, est une extension du langage CLP($\mathcal{FD}$) (version ancêtre de GNU-Prolog) permettant de valuer les contraintes dans des demi-anneaux, de façon à modéliser les contraintes floues, les problèmes sur-contraints, ainsi que la satisfaction approchée de contraintes.

5.5 CLPGUI

Participant : François Fages [correspondant].

CLPGUI est une interface graphique écrite en Java qui permet de visualiser et piloter l'exécution d'un programme CLP($\mathcal{FD}$), en l'occurrence GNU-Prolog.

La visualisation est centrée sur une représentation 3D du domaine des variables au cours du temps. Les interactions sont fondées sur un modèle d'exécution incrémental qui permet d'ajouter de nouvelles contraintes ou de nouveaux buts au cours de l'exécution, et de garder trace des retours en arrière.

Cette interface graphique sera étendue pour gérer les traces génériques d'exécution et de propagation des contraintes qui sont en cours de définition dans le projet RNTL OADymPPac.

L'implémentation réalisée est disponible à l'URL `http://contraintes.inria.fr/~fages/CLPGUI/`. Une version préliminaire de cette interface avait été développée cet été sur le Work Bench de réalité virtuelle par Jean-Michel Leconte avec le concours du projet I3D.

5.6 ISO Prolog

Participant : Pierre Deransart [correspondant].

Pierre Deransart, en collaboration avec AbdelAli Ed-Dbali, de l'Université d'Orléans, maintient un système de pages WEB interactives pour la norme ISO Prolog, développé par J. Hodgson. Ces pages permettent d'une part de consulter la documentation et les références de la

norme ISO Prolog, et d'autre part de faire passer des batteries de tests en ligne utilisant la spécification formelle du standard ISO Prolog `http://pauillac.inria.fr/~hodgson/prolog/`.

6 Résultats nouveaux

6.1 Langages concurrents avec contraintes et logique linéaire

Participants : François Fages, Sylvain Soliman.

La traduction des programmes CC en logique linéaire a été raffinée par une nouvelle traduction des systèmes de contraintes qui permet de caractériser exactement l'ensemble des succès des calculs CC (et non pas l'ensemble de leurs conséquences logiques) [4, 31]. Ces résultats ont permis de lever toutes les restrictions concernant la caractérisation des suspensions CC en logique non-commutative.

D'autre part, une implémentation restreinte d'une méthode originale de vérification par *phase model checking* a été réalisée par Sylvain Soliman en GNU-Prolog [22, 4]. Cette implémentation utilise les contraintes arithmétiques pour la recherche d'espaces de phases, singletons ou de faible cardinalité, sur le monoïde des nombres entiers naturels. Elle a permis de prouver automatiquement des propriétés de sureté de fonctionnement de programmes simples de protocoles de communication. Les travaux vont se poursuivre sur le traitement d'exemples de programmes plus complexes en explorant la recherche d'autres espaces de phases par des techniques de propagation de contraintes.

6.2 Typage des programmes logiques avec contraintes

Participants : Emmanuel Coquery, Pierre Deransart, François Fages.

Il est possible de typer statiquement les programmes logiques avec un système de types à la ML avec polymorphisme paramétrique, mais cela ne permet pas de rendre compte des coercitions entre domaines de contraintes (ex. booléens comme nombres entiers, listes comme termes, etc.), et ne permet bien sûr pas de typer les prédicats de métaprogrammation qui sont largement utilisés pour la programmation des procédures de recherche.

Notre approche de ces problèmes est fondée sur la conception d'un système de types avec sous-typage suffisamment souple pour autoriser les coercions entre domaines de contraintes, ainsi que le typage des prédicats de métaprogrammation [8, 12]. Les propriétés de stabilité du typage par réduction ont été établies, et l'implémentation réalisée par Emmanuel Coquery a permis de valider, sur un plan pratique, les hypothèses de ce système de types pour différentes versions de Prolog avec contraintes (ISO-Prolog, GNU-Prolog et SICStus Prolog). Ainsi les bibliothèques de SICStus-Prolog (3000 prédicats) ont été typées avec des temps d'exécution satisfaisants (1 seconde à 5 minutes) pour la vérification des types, ainsi que pour l'inférence du type des prédicats (à l'exception toutefois de trois bibliothèques contenant des composantes d'appels mutuellement récursifs de très grande taille, pour lesquelles l'inférence du type des prédicats nécessite 1 heure de calcul).

Notons que le système Wallace, développé par François Pottier dans le projet Cristal, est utilisé pour la résolution des inéquations de sous-typage. L'algorithme de résolution a cependant

été réimplémenté cette année dans le langage des *Constraint Handling Rules* (CHR) [11] de façon à disposer d'une implantation complète en Prolog facilement intégrable et pouvant être utilisée pour tester à l'exécution la satisfiabilité des systèmes de contraintes typés. De façon inattendue, l'implémentation dans CHR s'est avérée nettement plus efficace que Wallace pour l'inférence du type des prédicats, qui ne nécessite plus que 5 minutes CPU pour les bibliothèques les plus difficiles. Ce gain de performances s'explique par la possibilité d'effectuer des opérations d'unification dans CHR qui réalisent des simplifications du système de contraintes de sous-typage plus tôt que dans Wallace. Cette nouvelle implémentation nous permet également, grâce au traitement du non-déterminisme dans CHR, d'explorer et d'expérimenter la résolution des inéquations de sous-typage dans des structures plus générales que des treillis (quasi-treillis et ordres partiels), dont la décidabilité est un problème ouvert.

L'étude du typage polymorphe simple des programmes logiques a par ailleurs été poursuivie. Dans [24, 14] nous avons montré que la propriété de « généricité de tête », habituellement admise pour le typage statique des clauses, n'était pas nécessaire pour assurer la stabilité du typage par réduction, en définissant des conditions plus générales (mais dont la vérification est aussi plus complexe). Ces travaux répondent à différentes préocupations. Peut-on concevoir des conditions plus générales que la « condition de tête », permettant de mieux rendre compte de situations de polymorphisme *ad hoc* ou évitant d'imposer, comme en ML, des conditions strictes dans les appels récursifs et donc d'avoir un système de types moins contraignant ? Des conditions relaxées permettent-elles d'obtenir plus aisément la stabilité du typage par réduction dans le cas du sous-typage ? Une approche plus générale aide-t-elle au rapprochement des visions différentes du typage prescriptif ou descriptif ? Quelques résultats d'indécidabilité ont été obtenus concernant différentes conditions assurant la stabilité du typage par réduction [6] qui contribuent à éclairer le premier point.

6.3 GNU-Prolog

Participants : Philippe Codognet, Daniel Diaz.

Durant cette année nous avons continué le développement du système GNU Prolog (8 nouvelles beta-versions et 1 nouvelle version stable). Plusieurs bogues ont été corrigés et l'efficacité du système a été améliorée d'un facteur 1,4 par rapport à la version stable précédente [7]. Ceci a été obtenu par une ré-implantation des types de données Prolog (nouveau schéma de marquage) et par la spécialisation de certaines fonctions souvent utilisées. Une version Win32 (pour machines à base de systèmes Windows98/ME/2000/NT/XP) est en cours, et une version bêta devrait voir le jour courant novembre. Pour développer ce système, nous avons soumis un projet de collaboration avec l'Université d'Evora au Portugal (collaboration INRIA-ICCTI) où GNU-Prolog est utilisé de manière intensive puisqu'il est au centre de la gestion du système d'information de l'Université (gestion des enseignants, étudiants, salles, etc.). Ce projet vise à étendre GNU Prolog comme suit :

- implantation de mécanismes de *multi-threading* ;
- ajout d'un environnement de programmation d'interfaces graphiques en GNU-Prolog ;
- implantation de la programmation contextuelle dans GNU-Prolog.

Ces points correspondent à une demande des utilisateurs (notamment la programmation

d'interfaces graphiques) et ouvriront de nouveaux horizons à GNU-Prolog.

6.4 Contraintes flexibles

Participant : Philippe Codognet.

De nombreux problèmes de résolution de contraintes ne peuvent être exprimés dans le cadre classique où chaque contrainte doit recevoir une valeur de vérité booléenne (vrai/faux). Il est en effet très difficile, voire impossible, de résoudre dans le cadre CSP ou PLC classique les problèmes sur-contraints (l'ensemble total des contraintes posées est insolvable, mais on cherche un sous-ensemble qui maximise le nombre de contraintes satisfaites), les problèmes utilisant des priorités ou des préférences explicites sur les contraintes, les problèmes flous, qui sont pourtant très nombreux dans les applications réelles. Nous avons donc développé depuis plusieurs années, en collaboration avec Francesca Rossi (Université de Padoue, Italie), un cadre général pour prendre en compte et résoudre efficacement de tels problèmes. Ce cadre théorique est celui des contraintes valuées dans des demi-anneaux : les contraintes deviennent donc des fonctions associant à un n-uplet un élément d'un demi-anneau.

Nous travaillons actuellement à développer des nouveaux algorithmes pour résoudre les contraintes flexibles dans le cadre des contraintes valuées sur les demi-anneaux. Ainsi nos recherches portent sur les techniques d'abstraction entre domaines de valuation, permettant d'effectuer un calcul plus simple dans le domaine abstrait, et d'intégrer les résultats du domaine abstrait dans le domaine concret en fournissant des bornes encadrant la valeur des solutions optimales concrètes. Ceci aboutit à de nouveaux types d'algorithmes pour résoudre par exemple les contraintes floues en utilisant un simple solveur booléen et des techniques d'abstraction. Une extension aux domaines continus est également en cours.

6.5 Contraintes globales

Participants : Daniel Diaz, François Fages, Rémy Haemmerlé, Alexis Saurin, Jean-Claude Sogno.

Cette année nous avons étendu GNU-Prolog pour traiter les contraintes linéaires sur les nombres réels suivant le schéma CLP($\mathcal{R}$) [26], en réalisant une interface avec l'implémentation de l'algorithme du Simplexe développée par Jean-Claude Sogno. Cette interface permet de traiter les contraintes linéaires sur les réels conjointement aux contraintes sur les domaines finis de GNU-Prolog. L'interface peut être réutilisée pour connecter GNU-Prolog à d'autres implémentations plus efficaces du Simplexe ou d'autres méthodes de résolution.

GNU-Prolog a également été étendu par une interface permettant de définir en langage C des contraintes globales [28]. La contrainte `fd_all_different(List)` utilisant l'algorithme de Régin (couplage maximal dans un graphe biparti) a été implémentée de cette façon dans GNU-Prolog. D'autres contraintes globales pourront désormais être introduites grâce à cette interface.

6.6 Contraintes linéaires diophantiennes

Participant : Jean-Claude Sogno.

La résolution d'un système linéaire diophantien (énumération des solutions minimales) est un problème difficile. La plupart des méthodes publiées traitent le problème dans sa formulation initiale, pouvant comporter équations et inéquations. Or, le coût de calcul de ces méthodes croît fortement avec le nombre de variables. Nous proposons de substituer un système réduit équivalent, composé uniquement d'inéquations, et de lui appliquer un algorithme capable de traiter les inéquations. Le sous-système composé des équations est résolu par une méthode classique (Hermite) fournissant une solution paramétrique avec moins de variables. Cette solution est reportée dans le sous-système d'inéquations d'origine. Le nombre de variables décroît d'une valeur égale au nombre d'équations. Le mode de calcul de la phase finale dépend de ce nombre de variables. Lorsqu'il ne dépasse pas deux (cas fréquent avec les problèmes comportant surtout des équations), nous présentons une méthode fondée sur l'enchaînement d'un petit nombre de changements unimodulaires de variables. Sinon nous proposons l'utilisation d'un algorithme pour inéquations. Cette stratégie a été appliquée en particulier pour des problèmes publiés dans la littérature. Des systèmes qui avaient nécessité plusieurs minutes de calcul ont pu être résolus en quelques millisecondes. Ce travail a été présenté à la conférence INFORMS [21].

6.7 Ordonnancement disjonctif de tâches

Participants : François Fages, Jean-Claude Sogno.

Les méthodes d'ordonnancement des systèmes de contraintes diophantiennes linéaires et disjonctives sont classiquement fondées, d'une part sur des techniques de propagation de contraintes réduisant l'intervalle des bornes de variables, d'autre part sur un algorithme de séparation-évaluation. Nous proposons d'augmenter l'efficacité de la méthode en faisant largement appel pour les deux techniques à la programmation linéaire en nombres entiers : les systèmes que l'on est amené à traiter, comportant essentiellement des coefficients unitaires, ont une structure quasi unimodulaire permettant une résolution très rapide. En outre, certains calculs effectués solidairement se font plus rapidement que d'une manière isolée. Notre étude a surtout porté sur l'algorithme de séparation-évaluation. Avec la programmation linéaire, nous effectuons des sondages pour estimer le branchement le plus approprié. Des problèmes de taille moyenne ont pu être résolus uniquement avec l'algorithme de séparation-évaluation, sans optimisation locale. Le travail intégrant les deux techniques est en cours.

6.8 Recherche adaptative

Participants : Philippe Codognet, Daniel Diaz, Luc Hernandez.

Depuis quelques années une nouvelle approche dans les techniques de résolution de contraintes a connu un succès grandissant, il s'agit des techniques de recherche locale. Ainsi, contrairement aux techniques globales complètes telle la conjonction de la cohérence d'arc suivie de l'énumération explicite des choix restants, la recherche locale est une méthode heuristique incomplète qui ne fait une recherche exhaustive que dans un voisinage d'une solution partielle.

Ces idées sont en fait assez anciennes (certains algorithmes de résolution de problèmes du type « voyageur de commerce » étaient fondés sur de tels algorithmes il y a plus de 30 ans) mais ont montré seulement récemment leur utilité dans des problèmes généraux de contraintes (voir par exemple les travaux sur les problèmes de satisfaction booléenne au milieu des années 90). Elles ont vocation à être appliquées sur des problèmes fortement combinatoires pour lesquels la taille de l'espace de recherche prohibe l'utilisation de techniques exhaustives.

Nous avons mis au point une méthode générale de résolution de contraintes sur les domaines finis basée sur de nouvelles techniques heuristiques de recherche locale, nomée Recherche Adaptative [10]. En effet, la formulation des problèmes sous forme de CSP permet la définition et l'utilisation de fonctions heuristiques plus précises qu'une unique fonction de coût globale. Après une implantation prototype de cette méthode en Java, une implantation optimisée en C a été réalisée. L'évaluation de celle-ci montre que cette méthode est très efficace et permet d'aborder des problèmes dont la taille ne permet pas la résolution par des techniques classiques de propagation de contraintes. Comparée à d'autres implantations fondées sur la recherche locale, comme Localizer++ par exemple, l'efficacité estt meilleure d'un ou de deux ordres de grandeur sur certains exemples. Une étude plus exhaustive des performances ainsi qu'une généricité plus grande dans l'implantation sont maintenant à développer.

Il est intéressant de noter que la Recherche Adaptative a été utilisée avec succès pour des applications de conception musicale et sonore, dans le cadre de la thèse de Charlotte Truchet à l'IRCAM[23].

6.9 Réconciliation

Participant : François Fages.

En collaboration avec Marc Shapiro du centre de recherche Microsoft à Cambridge, nous étudions un problème combinatoire de réconciliation d'actions dans les applications nomades. Pour ce problème montré NP-difficile deux prototypes ont été développés [17] : l'un en programmation logique avec contraintes booléennes et entières, qui permet de prouver l'optimalité des solutions jusqu'à quelques centaines de transactions, l'autre procédant par recherche adaptative. Les travaux se poursuivent notamment sur la complexité pratique de ce problème [25], sur les techniques d'énumération heuristique, et sur la méthode de recherche adaptative qui, bien que non compétitive actuellement sur ce problème, semble prometteuse pour résoudre les problèmes de réconciliation en ligne.

6.10 Contraintes 3D et Réalité Virtuelle

Participants : Philippe Codognet, Nadine Richard, Yoann Fabre.

Le projet InViWo (Intuitive Virtual Worlds), qui a fait l'objet de la thèse de Nadine Richard, a pour objectif de concevoir et de mettre en œuvre un outil de création de mondes virtuels, et plus particulièrement de proposer un langage de description de comportements d'agents évoluant dans des environnements virtuels en 3D [20, 3]. Nous considérons dans ce cadre un monde virtuel habité comme étant un système multi-agent homogène, dans lequel certains agents (les avatars) peuvent être contrôlés par des utilisateurs. Nous proposons le modèle

d'agent InViWo à la fois générique et dynamique ainsi qu'une architecture de sélection de l'action distribuée associée à un mécanisme d'arbitrage permettant de combiner les décisions prises par les différents modules comportementaux constituant le processus de décision d'un agent virtuel. Le réseau dynamique formé par les modules comportementaux fonctionne selon le modèle réactif synchrone, en s'inspirant en particulier du langage ESTEREL. Pour décrire la partie réactive des modules comportementaux, nous avons défini le langage MARVIN, proche d'ESTEREL, fondé sur des constructions spécifiques permettant de décrire entièrement un agent InViWO.

La bibliothèque de programmation et d'exécution des agents a été réalisée dans l'environnement Java et Java3D, et une plateforme prototype a été implémentée pour permettre de valider l'approche.

Une nouvelle voie de recherche a été ouverte avec la thèse de Yoann Fabre, centrée sur les problèmes de Réalité Virtuelle distribuée.

6.11 Environnements de mise au point

Participants : Pierre Deransart, Daniel Diaz, François Fages, Gérard Ferrand, Ludovic Langevine, Jean-Michel Leconte.

Outre la publication de [15, 16], le travail sur les environnements de mise au point a principalement porté sur la définition et la mise en place d'une approche nouvelle du débogage de performance pour la programmation avec contraintes (PaC).

L'idée générale est illustrée par la figure 1 : différents solveurs sont susceptibles d'être observés par différents outils, par exemple un outils de *profiling* indiquant les contraintes les plus sollicitées, des outils de représentation de l'espace de recherche sous forme d'un arbre (projection verticale) ou sous forme d'un *starfield* (projection pondérée horizontale). On recherche donc une méthode qui permette un développement d'outils de manière indépendante des plateformes ainsi que leur réutilisation sur différentes plateformes.

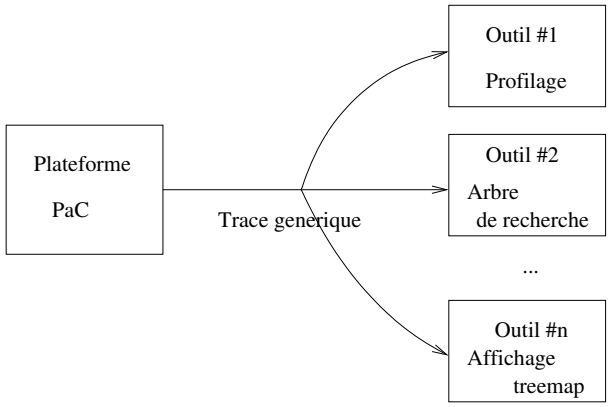

FIG. 1 – Emission d'une trace générique

Le problème est ouvert dans la mesure où il n'existe pas de méthode de traçage qui permette

d'observer la partie la plus délicate et complexe des solveurs, à savoir la propagation. Il n'existe pas non plus d'outils aisément portables sur différentes plateformes.

Trois questions principales doivent être résolues dans ce but : production d'une trace « générique », définition d'un format d'échange entre le traceur et les outils, définition des outils eux-mêmes.

6.11.1 Trace générique

La méthode que nous avons choisie pour essayer d'approcher une trace générique est détaillée dans [27] et provisoirement publiée dans [19]. Elle consiste à définir les traceurs à partir d'une sémantique formelle des solveurs, à implanter des prototypes permettant d'extraire des informations et à tester quelques outils. Cette démarche nous permet d'obtenir rapidement une expertise sur la nature des « bonnes » traces.

Pour la partie « modèle de solveurs », l'étude est à la fois théorique et empirique. L'approche théorique est considérée en collaboration avec nos collègues d'Orléans. Ceux-ci ont proposé ([18]) d'intégrer, dans le cadre unificateur des itérations chaotiques qui décrit la sémantique opérationnelle de solveurs à base de règles, une représentation abstraite de la structure de données qui permet de détecter un point fixe, ainsi que le mécanisme de recherche utilisé pour trouver toutes les solutions.

Sur un plan pratique, les étapes principales de calcul du point fixe utilisant les opérateurs de réduction sont repérées par des événements particuliers (appelés *ports*). Un premier modèle « déclaratif »a été décrit, susceptible d'être facilement implanté. Les événements comportent un grand nombre d'attributs qui garantissent que toutes les informations utiles sont potentiellement accessibles.

La partie « traceur » est fondée sur un méta-interprète écrit en Prolog (norme ISO), implanté provisoirement sur ECLiPSe et sur un module simplifié « à la Opium », facile à implanter et qui permet d'analyser la trace et d'en extraire toutes informations utiles.

Finalement, quelques outils sont développés, et ceux des partenaires du projet OADymPPaC sont testés, de manière à mieux comprendre les concepts utiles efficacement observés à l'aide de ces outils.

Cette méthodologie qui permet de tester toutes sortes de traces et d'outils, bien qu' expérimentalement limitée à de petits exemples, se révèle suffisamment utile en pratique pour être poursuivie.

6.11.2 Format d'échange

Une fois un modèle de trace défini, l'ensemble des partenaires du projet OADymPPaC a défini un format concret permettant l'échange entre les solveurs et les outils, fondé sur une DTD XML [30].

La trace produite, très volumineuse, n'est en principe pas destinée à être stockée dans un fichier XML, mais à être envoyée à la volée à un programme qui l'exploitera (analyseur ou outil de visualisation). Afin de pouvoir l'analyser efficacement nous envisageons l'utilisation du format WBXML et de bibliothèques de codage/décodage.

6.11.3 Concepts et outils

La trace ne peut être conçue de façon suffisamment générale qu'en prenant en compte le plus grand nombre d'outils potentiellement utiles pour l'analyse des phénomènes que l'on souhaite observer. L'objectif est en effet d'obtenir la plus grande panoplie d'outils susceptibles d'aider à résoudre des problèmes avec contraintes.

Nous avons donc commencé à conduire des expérimentations avec les partnaires industriels du projet OADymPPaC. Plus spécifiquement, nous avons commencé à étudier quelques outils qui utilisent des informations obtenues à l'aide du méta-interprète et de l'analyseur.

Un premier travail a porté sur la visualisation 3D de l'évolution des domaines de variables. Ce travail est décrit dans [29]. Il se situe dans la continuation d'outils développés dans le cadre du projet DiSCiPl (TRIFID). Il est possible d'observer l'effet de stratégies particulières sur un très grand nombre d'évènements.

L'interface graphique CLPGUI est une autre contribution du projet qui, au delà de la visualisation 3D, permet d'interagir avec le programme CLP en ajoutant incrémentalement de nouvelles contraintes ou de nouveaux buts, et en pilotant les retours en arrière.

6.12 Preuves de programmes par co-induction

Participants : Sorin Craciunescu, François Fages.

Un système de preuve par induction et co-induction a été développé pour prouver la correction des programmes logiques avec contraintes [13]. Ce système utilise le principe d'induction pour prouver l'inclusion des ensembles de succès de deux programmes logiques avec contraintes, et un principe symétrique de co-induction pour prouver l'inclusion des échecs finis des programmes. Cette étude se fait en collaboration avec Dale Miller et Jérémie Wajs de Penn State College University.

7 Contrats industriels (nationaux, européens et internationaux)

7.0.1 Problèmes de réconciliation à base d'historiques dans les applications nomades

Participant : François Fages.

Un contrat de collaboration de recherche a été mis en place avec le centre Microsoft Research de Cambridge (équipe de Marc Shapiro) sur les problèmes combinatoires de réconciliation à base d'historiques dans les applications nomades. La recherche porte sur la modélisation des problèmes de reconciliation, leur complexité, et sur les méthodes exactes (programmation logique avec contraintes) et approchées (recherche locale ou adaptative) pour les résoudre.

8 Actions régionales, nationales et internationales

8.1 Actions nationales

8.1.1 Projet OADymPPaC

Participants : Pierre Deransart, François Fages, Gérard Ferrand, Ludovic Langevine, Philippe Codognet, Daniel Diaz, Jacob Navia.

Le projet RNTL OADymPPaC (Outils pour l'Analyse Dynamique et la mise au Point de Programmes avec Contraintes) a débuté le 14 novembre 2000 pour une durée de 3 ans.

Il regroupe deux partenaires industriels (Cosytec et ILOG) et quatre partenaires académiques (INRIA-Rocquencourt, Université d'Orléans, Ecole des Mines de Nantes et INSA/IRISA).

Le projet est organisé en cinq sous-projets. Nous décrivons ici brièvement l'implication de notre équipe dans chacun d'eux. Un rapport complet sera édité [33].

- SP1 Modèles conceptuels et extraction de traces
  Ce sous-projet porte sur deux aspects fondamentaux pour le projet, tant du point de vue théorique que pratique. Le premier aspect traite de la modélisation des solveurs de contraintes afin d'obtenir une description uniforme, indépendante du domaine de contraintes, de leur comportement qui puisse servir de modèle de base général pour les outils de mise au point. Le second aspect concerne l'extraction et l'analyse de traces, goulot d'étranglement auquel tout outil est préalablement soumis. Pour l'extraction de trace, il s'agit de définir la collection d'événements génériques qui doivent être extraits de tout solveur pour analyser son comportement. Pour l'analyse, il s'agit de définir un langage commun permettant de spécifier les informations pertinentes à extraire pour un outil donné.
  Notre effort s'est essentiellement concentré sur la partie « traces » en collaboration avec Mireille Ducassé et Erwan Jahier de l'IRISA/INSA. Nous avons défini une méthode, réalisé une première implantation sur EcliPse et effectué les premiers tests (voir section 6.11). L'achèvement de cette partie est fondamental pour la suite des travaux de l'ensemble des partenaires. Cette approche méthodologique et ce premier prototype nous permettent, ainsi qu'à d'autres partenaires, de produire et d'échanger des traces « test » à partir desquelles chacun pourra développer et échanger ses expériences.
  Notre objectif à terme est de disposer d'un traceur/analyseur sur GNU-Prolog afin de réaliser et de pouvoir utiliser plusieurs outils de déboggage sur la plateforme.
- SP2 Modèles de visualisation
  Ce sous-projet sert de pivot à l'ensemble du projet : il s'agit, en partant de l'expérience et des besoins en programmation avec contraintes de tous les partenaires, de parvenir à dégager les concepts essentiels et à définir le langage minimal qui permettra aux solveurs et aux outils de communiquer. Les outils concernés sont essentiellement des outils de visualisation fortement paramétrisables (une même information peut être visualisée sous différentes formes à la demande de l'utilisateur). L'objectif est de pouvoir « regarder » les traces de manière à en visualiser les caractères les plus utiles pour la compréhension des effets des stratégies de résolution utilisées dans les solveurs.
  Dans cette première phase du projet nous nous sommes concentrés, avec Jean-Daniel

Fekete de l'EMN et les autres partenaires, sur la définition d'un format concret d'émission de traces. Ce format est décrit sous forme d'une DTD XML [30]. Un rapport à venir traitera des principaux concepts utiles à l'analyse du comportement des solveurs et susceptibles de faire l'objet de visualisation.

- SP3 Analyse de performance et spécification de vues
  L'un des principaux concepts à visualiser est l'arbre de recherche, représentation de l'espace de recherche de la ou des solutions. Dans le projet DiSCiPl `http://discipl.inria.fr`, de nombreux outils destinés à analyser cet espace ont été étudiés. Nous avons commencé à analyser d'autres formes de visualisation, en collaboration avec Cosytec.
- SP4 Techniques et composants génériques de visualisation
  Le sous-projet SP4 traite du problème de la visualisation des traces comme outil d'analyse dynamique d'un point de vue général et de l'application de certaines techniques à la mise au point de programmes avec contraintes en particulier. Dans ce cadre nous avons commencé une collaboration avec ILOG afin de produire des traces sur lesquelles les outils de ILOG-View pourront être expérimentés.
- SP5 Administration du projet et dissémination
  Notre équipe est coordinatrice du projet. A ce titre nous avons contribué activement à l'organisation du projet et à la dissémination des résultats intermédiaires de la manière suivante (outre les publications citées en bibliographie).
  - Création et animation du site WEB du projet `http://contraintes.inria.fr/OADymPPaC` ainsi que de la liste de diffusion électronique. Le site sert d'une part à la diffusion des résultats publics et à l'organisation de la vie interne du projet (intranet).
  - Organisation de la réunion de démarrage (26 Janvier 2001) et participation avec un stand (organisé avec M. Ducassé) aux journées RNTL du 26-27 avril 2001.
  - Lancement et co-organisation de plusieurs réunions thématiques sur la modélisation (EMN, Nantes, 3 mars 2001), traces et visualisation (Cosytec, Orsay, 20 juin 2001), format générique de traces (LIP6, Paris, 27 septembre 2001).
  - Edition d'une affiche et d'une *fact sheet* de présentation du projet.
  - Présentation du projet OADymPPaC à l'occasion de diverses réunions dont le *workshop « Analysis and Visualization of Constraint Programs and Solvers »* à CP'2000 (22 septembre 2000) le *Chip-users club* (22 novembre 2001) et le *workshop* WLPE à CP/ICLP'2001 (1 décembre 2001).

8.1.2 Autres collaborations nationales

Des liens étroits existent avec le DI de l'ENS Ulm Paris (M. Fernandez), l'IRIN de Nantes (F. Benhamou, F. Goualard, L. Granvillier), le LIFO d'Orléans (G. Ferrand, A. Ed-Dbali, A. Lallouet, A. Tessier), le LIM de Marseille (A. Colmerauer, M. Van Caneghem), le LMD de Marseille (J.Y. Girard, P. Ruet), le projet Protheo de l'INRIA (C. et H. Kirchner, P.E. Moreau).

8.2 Actions européennes

8.2.1 TMR Linear Logic in Computer Science

Participants : François Fages, Sylvain Soliman.

Nous participons au réseau européen TMR *Linear Logic in Computer Science* coordonné par le site de Marseille (J.Y. Girard), avec les sites de Paris (V. Danos), Bologne (A. Asperti), Cambridge (M. Hyland), Edinburgh (S. Abramsky), Lisbonne (J.L. Fiadeiro), et Rome (M. Abrusci).

8.2.2 Projet de l'institut Franco-Russe Liapunov

Participants : Frédéric Benhamou, Philippe Codognet, François Fages.

Ce projet reconduit de l'Institut Liapunov concerne une collaboration entre l'Institut d'intelligence artificielle de l'Académie des sciences de Novossibirsk, l'Université de Nantes, l'INRIA et l'Université d'Orléans. Cette collaboration porte essentiellement sur les langages de contraintes sur les nombres réels et sur la résolution de contraintes par méthodes d'intervalles. Elle permet des échanges de chercheurs et d'étudiants.

8.2.3 Centre de Coopération universitaire Franco-Bavarois

Participants : Sorin Craciunescu, François Fages.

Une subvention du centre de coopération universitaire Franco-Bavarois a été obtenue pour l'étude des méthodes de preuve par co-induction d'équivalence de programmes CHR, en collaboration avec François Bry, Thom Fruehwirth et Slim Abdennadher de l'Université Ludwig-Maximilians de Munich.

8.2.4 Projet Galilée

Participants : Philippe Codognet, Daniel Diaz.

Une subvention du Ministère des Affaires Etrangères a été obtenue dans le cadre du programme Galilée de coopération bilatérale avec l'Italie. Cette coopération concerne l'Université Paris 6 (Philippe Codognet) et l'Université de Padoue (Francesca Rossi) sur le thème de la résolution de contraintes flexibles et du cadre théorique des contraintes valuées sur des demi-anneaux.

8.2.5 ERCIM working group on Constraints

Philippe Codognet a co-fondé le *ERCIM working group on Constraints* avec Krzysztof Apt (CWI, Amsterdam), qui en est le coordinateur. Ce réseau regroupe des équipes à l'INRIA, CWI (Hollande), CNR (Italie), GMD (Allemagne), les Universités de Prague et Brnó (République Tchèque).

8.2.6 Portugal

Pierre Deransart est responsable avec Marie-Christine Imbert (DRI), puis actuellement Martine Le Corre des relations avec le Portugal. Un appel à propositions INRIA/ICCTI pour des projets franco-portugais (octobre 2001) ainsi que leur sélection ont été réalisés.

8.3 Actions internationales

8.3.1 NSF-CNRS research collaboration, Penn State College, ENS Ulm, INRIA

Participants : Sorin Craciunescu, François Fages, Sylvain Soliman.

François Fages a initié un contrat de collaboration NSF-CNRS, auquel nous continuons de participer, sur « Spécifications logiques et outils de vérification pour les langages concurrents », avec l'Université de Penn State College (D. Miller, C. Palamidessi) et l'ENS Ulm (M. Fernandez). Un *workshop* sur ce thème a été organisé à l'INRIA en mai.

8.3.2 Brésil

Participant : Pierre Deransart.

Pierre Deransart est responsable avec Marie-Christine Imbert (Direction des Relations Internationales) des relations avec le Brésil. A ce titre il a coordonné l'appel à propositions INRIA/CNPq pour des projets franco-brésiliens ainsi que leur sélection. Il a également participé au 3^{ieme} séminaire d'évaluation des projets organisé par le CNPq à Rio de Janeiro (octobre 2001).

8.4 Visites, et invitations de chercheurs

Nous avons accueilli pour de courtes visites les chercheurs suivants : Slim Abdennadher (Université Ludwig Maximilian, Münich, Allemagne), Dale Miller, Catuscia Palamidessi et Jérémie Wajs (CSE, Penn State College, USA), Frank Valencia (Aarhus, Danemark).

9 Diffusion de résultats

9.1 Animation de la Communauté scientifique

9.1.1 Comités éditoriaux de revues

Philippe Codognet est membre du comité de rédaction de la revue *ACM Transactions on Computational Logic* (ACM Press) et de la revue *Constraints, an International Journal* (Kluwer Academic Press).

9.1.2 Comités de programmes de conférences

Frédéric Benhamou a fait partie des comités de programme des conférences NimesTIC'2001, (Nîmes, France), PSI'2001 (*Perspectives of Systems Informatics*, Akademgorodok, Russie), et ICLP'2001 (*International Conference on Logic Programming*, Paphos, Chypre).

Philippe Codognet a été président des comités de programme de ICLP2001, *16th International Conference on Logic Programming* (Chypre, novembre 2001), et des JFPLC'2001, les Dixèmes Journées Francophones de Programmation Logique et Programmation par Contraintes (Paris, avril 2001). Il a été membre du comité de programme de CP'2001, *Constraint Programming* (Chypre, novembre 2001).

Pierre Deransart a été membre du comité de programme de JFPLC'2001 et de WLPE'2001 (*11th Workshop on Logic Programming Environments*, Paphos, Chypre, 1^{er} décembre 2001). Il est membre des comités de programme de FLOPS'2002 (*Sixth International Symposium on Functional and Logic Programming*), des JFPLC'2002 (Nice, juin 2002).

François Fages a été membre du comité de programme des JFPLC'2001 (Paris, avril 2001) et du *Workshop Musical Constraints* associé à l'*International Conference on Principles and Practice of Constraint Programming* CP'2001 (Chypre, novembre 2001). Il est membre du comité de programme des JFPLC'2002 (Nice, mai 2002), de ICLP'2002, *17th International Conference on Logic Programming* (Copenhague, juillet 2002) et de PPDP'2002, *Principles and Practice of Declarative Programming* (Pittsburgh, septembre 2002).

Gérard Ferrand a été membre du comité de programme de JFPLC'2001.

9.1.3 Organisation de manifestations

Philippe Codognet a été membre du comité d'organisation d'ASTI'2001 (Paris, avril 2001).

Pierre Deransart a été Président et organisateur des JFPLC'2001 co-organisé avec ASTI'2001 à Paris. Il a été membre du comité de pilotage de ASTI'2001.

François Fages est membre du comité de pilotage de la conférence ACM-Sigplan *Principle and Practice of Declarative Programming*, PPDP. Il a été membre du comité d'organisation des Premieres Rencontres des Sciences et Technologies de l'Information à la Cité des Sciences et de l'Industrie, ASTI'2001 (Paris, avril 2001), et responsable des expositions présentées à la Cité des Sciences à cette occasion.

9.1.4 Jurys

François Fages a été rapporteur de la thèse d'habilitation de Slim Abdennadher (Université Ludwig Maximilian de Münich), et de la thèse de doctorat de Hubert Dubois (LORIA, Nancy), Il a participé au jury de la thèse de Sylvain Soliman (Université Denis Diderot, Paris 7), Il a été membre du jury du challenge de programmation ROADEF'2001 de l'Association Française de Recherche Opérationnelle et d'Aide à la Décision. Il est membre du jury d'admission du concours CR2 de l'INRIA Rocquencourt, du conseil scientifique de l'Institut Liapunov, et des commissions de spécialistes des Universités Denis Diderot, Paris 7, et Paris Sud, Paris 11.

9.1.5 Responsabilités associatives

Philippe Codognet est membre du comité exécutif de l'ALP, *International Association for Logic Programming*.

Pierre Deransart est secrétaire général de l'Association Française pour la Programmation en Logique et la programmation par Contraintes (AFPLC) et membre du conseil des associations de l'ASTI.

François Fages est président de l'AFPLC, et était trésorier de l'Association Française des Sciences et Technologies de l'Information (ASTI) jusqu'à cette année.

9.2 Enseignement

Les doctorants du projet participent à divers enseignements en premier et second cycles universitaires. Nous détaillons ici les enseignements de troisième cycle.

9.2.1 DEA Sémantique, preuves et programmation, Universités Paris 6, Paris 7, Paris 11, ENS Ulm, ENS Cachan, Ecole Polytechnique, CNAM

Philippe Codognet et François Fages font un cours sur la « programmation par contraintes » dans la filière « langages de programmation » de ce DEA.

9.2.2 DEA IARFA, UNiversité Paris 6

Philippe Codognet donne un cours de « méthodes et techniques de résolution de contraintes » dans le DEA IARFA (Intelligence Artificielle et Reconnaissance des Formes).

9.2.3 DEA Informatique, Université d'Orléans

Pierre Deransart, François Fages et Gérard Ferrand font un cours de « Programmation en Logique et par Contraintes » dans ce DEA.

9.2.4 DESS GLA, Université de Paris 6

Philippe Codognet donne un cours de « programmation par contraintes » dans le DESS Genie des Logiciels Applicatifs.

9.2.5 ESSLLI 2001, Helsinki, Finlande

François Fages a donné un cours sur la Programmation Logique avec Contraintes [32] dans la filière *Logic and Computation* de la 13ième *European Summer School in Logic, Language and Information* ESSLLI'2001.

10 Bibliographie

Livres et monographies

[1] P. Codognet, (éditeur), *Logic Programming, proceedings of ICLP-01, 17th International Conference on Logic Programming*, Springer Verlag, novembre 2001.

[2] P. Codognet, (éditeur), *Programmation en Logique avec Contraintes, actes de JFPLC'2001, 10èmes journées francophones de programmation en lofique et de programmation par contraintes*, Hermes, avril 2001.

Thèses et habilitations à diriger des recherches

[3] N. RICHARD, *Description de comportements d'agents autonomes évoluant dans des mondes virtuels*, thèse de doctorat, Ecole Nationale Supérieure des Télécommunications, octobre 2001.

[4] S. SOLIMAN, *Programmation Concurrente avec Contraintes et Logique Linéaire*, thèse de doctorat, Université Denis Diderot, Paris 7, avril 2001.

Articles et chapitres de livre

[5] P. CODOGNET, F. PACHET, (ÉDITEURS), « Special issue on Constraints for multimedia applications », *CONSTRAINTS, an International Journal 6*, 1, janvier 2001.

[6] P. DERANSART, J.-G. SMAUS, « Subject Reduction of Logic Programs as Proof-Theoretic Property », *in : The Journal of Functional and Logic Programming*, Electronic Journal published by the EAPLS, 2001, version étendue du papier FLOPS'2001, à paraître.

[7] D. DIAZ, P. CODOGNET, « Design and Implementation of the GNU Prolog System », *Journal of Functional and Logic Programming 2001*, 6, octobre 2001.

[8] F. FAGES, E. COQUERY, « Typing Constraint Logic Programs », *Theory and Practice of Logic Programming 1*, 6, novembre 2001, p. 751–777.

[9] F. FAGES, P. RUET, S. SOLIMAN, « Linear concurrent constraint programming : operational and phase semantics », *Information and Computation 165*, 2001, p. 14–41.

Communications à des congrès, colloques, etc.

[10] P. CODOGNET, D. DIAZ, « Yet Another Local Search Method for Constraint Solving », *in : proceedings of SAGA'01, 1st International Symposium on Stochastic Algorithms : foundations and applications*, K. Steinhöfel (éditeur), Springer Verlag, LNCS 2264, Berlin, Allemagne, décembre 2001.

[11] E. COQUERY, F. FAGES, « From Typing Constraints to Typed Constraint Systems in CHR », *in : Proceedings of the Third workshop on Rule-based Constraint Reasoning and Programming, associated to Constraint Programming CP'2001*, Chypre, novembre 2001.

[12] E. COQUERY, F. FAGES, « Programmes logiques avec contraintes typés : vérifer les coercions de domaines et la métaprogrammation à travers le sous-typage », *in : Actes des 10ièmes journées francophones de programmation en logique et de programmation par contraintes JFPLC'01*, Hermès (éditeur), p. 223–238, Paris, avril 2001.

[13] S. CRACIUNESCU, « Preuves de programmes logiques par induction et co-indutction », *in : Actes des 10ièmes journées francophones de programmation en logique et de programmation par contraintes JFPLC'01*, Hermès (éditeur), p. 287–302, Paris, avril 2001.

[14] P. DERANSART, J.-G. SMAUS, « Well-Typed Logic Programs Are not Wrong », *in : Fifth International Symposium on Functional and Logic Programming, FLOPS'2001, Tokyo, Japan*, H. Kuchen, K. Ueda (éditeurs), *LNCS*, 2024, Springer-Verlag, p. 280–295, mars 2001.

[15] A. ED-DBALI, M. A. S. BIGONHA, P. DERANSART, J. DE SIQUEIRA, R. DA BIGONHA, « HyperPro : an integrated documentation environment for CLP », *in : WLPE'2001, post conférence workshop de ICLP'2001, Chypre*, T. Kusalick (éditeur), p. 6, novembre 2001.

[16] A. ED-DBALI, M. A. S. BIGONHA, P. DERANSART, J. DE SIQUEIRA, R. DA BOGONHA, « HyperPro : un environnement intégré de documentation pour la PLC », *in : JFPLC'2001, Paris*, P. Codognet (éditeur), Hermès, p. 257–270, avril 2001.

[17] F. Fages, « CLP versus LS on log-based reconciliation problems », *in : Proceedings of the 6th ERCIM Workshop on Constraints*, Prague, République Tchèque, juin 2001.

[18] A. Lallouet, G. Ferand, J. Arsouze, « Chaotic derivations : an operational semantics for complete constraint solving », *in : RCORP'01, post ICLP/CP'01 workshop on Rule-Based Constraint Reasoning and Programming*, S. Abdennadher, T. Frühwirth (éditeurs), Paphos, Cyprus, 2001, `http://www.pms.informatik.uni-muenchen.de/ereignisse/iclpr2001.html`.

[19] L. Langevine, P. Deransart, M. Ducasse, E. Jahier, « Prototyping CLP(FD) tracers : a trace model and an experimental validation environment », *in : WLPE'2001, post conférence workshop de ICLP'2001, Chypre*, T. Kusalick (éditeur), p. 4, novembre 2001.

[20] N. Richard, P. Codognet, A. Grumbach, « The InViWo Toolkit : Describing Autonomous Virtual Agents and Avatars », *in : proceedings of IVA'2001, Intelligent Virtual Agents*, R. A. A. de Antonio, D. Ballin (éditeurs), Springer Verlag, LNCS 2190, Madrid, Espagne, septembre 2001.

[21] J.-C. Sogno, « The Shrinking Strategy for Solving Linear Diophantine Systems », *in : INFORMS International HAWAII*, Maui, Hawaii, juin 2001.

[22] S. Soliman, « Phase Model Checking for some Linear Logic Calculi », *in : Proceedings of the 2nd International Workshop on the Implementation of Logics IWIL'01*, Cuba, décembre 2001.

[23] C. Truchet, P. Codognet, G. Assayag, « Visual and Adaptive Constraint Programming in Music », *in : proceedings of ICMC'2001, International Computer Music Conference*, La Havane, Cuba, septembre 2001.

Rapports de recherche et publications internes

[24] P. Deransart, J.-G. Smaus, « Well-Typed logic programs are not wrong », *RR n°4082*, INRIA, décembre 2000, `http://www.inria.fr/rrrt/rr-4082`.

[25] F. Fages, « A constraint programming approach to log-based reconciliation problems », *rapport de recherche*, INRIA, août 2001.

[26] R. Haemmerlé, « Implémentation d'un solveur de contraintes linéaires dnas GNU-Prolog (annexes : "GNU Prolog CLP(R) : User Manual and Technical Manual") », *rapport de recherche*, INRIA. Rapport de stage de l'ESIEE Paris, août 2001.

[27] L. Langevine, P. Deransart, M. Ducasse, E. Jahier, « Tracing and Analysing execution of CLP(FD) Programs : a trace model and an experimental validation environment », *Rr*, INRIA, novembre 2001, Délivrable D1.2.1, à paraître en rapport INRIA, `http://contraintes.inria.fr/OADymPPaC/`.

[28] A. Saurin, « L'implantation des contraintes globales en GNU-Prolog (annexes : "Global constraints in GNU Prolog : User Manual and Technical Manual") », *rapport de recherche*, INRIA. Rapport de stage du magistère MMFAI de l'ENS Ulm., septembre 2001.

[29] R. Zoumman, « Analyse du comportement du solveur de contraintes basée sur la visualisation de traces », *Rapport de deuxième année, ISIMA, Clermont-Ferrand*, INRIA, septembre 2001.

Divers

[30] R. Debruyune, J.-D. Fekete, N. Jussien, M. Ghoniem, P. Deransart, L. Langevine, e et al., « Proposition de format concret pour les traces générées par les solveurs de contraintes », octobre 2001, Délivrable D2.2.2.1, `http://contraintes.inria.fr/OADymPPaC/Public/delivrables.html`.

[31] F. FAGES, S. SOLIMAN, « A precise logical semantics of concurrent constraint programs », février 2001, Soumis à publication.

[32] F. FAGES, « Constraint Logic Programming », Helsinki, Finlande, 2001, Lecture notes, ESSLLI'2001, Logic and Computation track.

[33] OADYMPPAC, « Premier rapport d'avancement », décembre 2001, édité par Pierre Deransart, à paraître.