INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Action 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]

Chercheur post-doctorant

Jan Smaus [du 1/1/2000 au 30/6/2000]

Ingénieur expert

Jean-Marie Geffroy [jusqu'au 1/3/2000 et depuis le 1/7/2000]

Chercheurs doctorants

Emmanuel Coquery [Bourse MESR, INRIA, depuis le 1/10/2000]

Sorin Craciunescu [AMX, INRIA]

Yoann Fabre [Bourse MESR, Paris 6, depuis le 1/10/2000]

Luc Hernandez [Bourse MESR, Paris 6]

Nadine Richard [Bourse INRIA, ENST, jusqu'au 31/08/2000]

Sylvain Soliman [Ingénieur DGA, INRIA]

Stagiaires

Emmanuel Coquery [du 1/4/2000 au 30/9/2000]

Mohamed Amin Kaid [du 1/4/2000 au 30/9/2000]

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 est une opération d'ajout de contrainte à la mémoire, 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 succés 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 drastique des coûts de développement et de maintenance du logiciel. Cette contribution de génie logiciel permet aussi d'attaquer des problèmes nouveaux, NP-difficile, 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 se situent 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 solveurs de contraintes, les langages de modélisation multiple, 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, que ce soit en termes d'expressivité, de complexité, de modèles d'exécution ou de coopération, ou 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 lequel 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 notoire 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 succés [3]. La traduction dans la logique non-commutative de Ruet-Abrusci permet de plus, sous certaines conditions, 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 des deux derniers points repose sur une extension naturelle des langages CC, nommée LCC, qui consiste simplement à considérer des systèmes de contraintes basés 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 problémes 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 accélerer 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 basé sur la sémantique logique des programmes, pour la conception de systèmes intéractifs 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) [2], impact des heuristiques etc...

Les résultats obtenus dans le projet encouragent la poursuite des recherches sur les méthodes de visualisation des contraintes et de l'espace de recherche, sur les méthodes de débogage basées 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, ingéniérie, transport, ordonnancement, allocation de ressources, placement, planification.

Les problèmes d'optimisation combinatoire rencontrés dans le monde industriel, et leur

impact économique, vont croissants, qu'il s'agisse de problèmes de:

- allocation de ressources;
- ordonnancement de tâches;
- placement;
- parallélisation;
- planification de personnel, rotation d'équipages;
- transport;
- 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, heuristiques, algorithmes sur les graphes.

L'apport de la programmation par contraintes se situe d'une part dans les techniques de consistance 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 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 suffisament 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 concurents avec contraintes, des solveurs de contraintes et de leurs implémentations. Les recherches sur LCC fournissent un cadre théorique pour traiter plusieurs aspects de la programmaion par contraintes qui ne se formalisent pas bien dans les cadres existants, en particulier pour la programmation des solveurs de contraintes spécifiques aux applications. 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 travail se developpe pour de nombreuses applications, du magasin virtuel pour le commerce électronique aux jeux vidéos. Concevoir et implanter des mondes virtuels restent cependant des tâches difficiles car peu d'outils de haut-niveau existent pour permettre 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 standards (Internet...) pose des problèmes en dehors du champ de l'informatique graphique 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 qui semble prometteuse et que nous étudions.

5 Logiciels

5.1 GNU-Prolog

Participants : Daniel Diaz [correspondant].

GNU-Prolog est un compilateur natif pour Prolog intégrant un puissant résolveur de contraintes sur les domaines finis. GNU-Prolog accepte donc des programmes Prolog avec contraintes et produit des executables (binaires) entièrement autonomes dont la taille reste réduite. GNU-Prolog offre également la possibilité d'exécuter un programme à l'aide de l'interpréteur pour en facilitier 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",... De plus il intègre un résolveur 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 est arrivée 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 15000 exemplaires ont été récupérés

sur le site FTP de l'INRIA (nous ne disposons pas de statistiques concernant le site FTP de GNU ni sur les nombreux miroirs de par le monde). GNU-Prolog est aujourd'hui inclus dans toutes les distributions officielles de Linux telles que: Debian, Mandrake, RedHat, ... Une discussion est en cours au sein de l'ALP (Association for Logic Programming) pour élire GNU-Prolog comme LE Prolog de l'association.

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 récupérer les dernières versions d'un module (CVS en mode lecture), proposer des modifications ou des "patchs", 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 lui donner les droits d'écriture sur le repositoire CVS. Nous espérons que cet effort contribuera à favoriser les développements extérieurs.

5.2 TCLP

Participants : Emmanuel Coquery [correspondant].

TCLP est un programme qui vise a typer les programmes écrits en Prolog et dans leurs extensions de programmation par contraintes [23]. Moyennant une description des types des différents constructeurs de termes et de prédicats, le système 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 basé 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 $list(\alpha) < term$ pour permettre de traiter les listes comme des termes.

L'implémentation réalisée est disponible à l'URL `http://www.mcs.anl.gov/Projects/autodiff/AD_Tools/` [24], elle utilise la librairie Wallace, développée par François Pottier dans le projet Cristal, pour la résolution des contraintes de sous-typage. Le système TCLP a été testé sur les prédicats prédéfinis de GNU-Prolog et Sicstus Prolog, sur toutes les librairies de Sicstus Prolog (plus de 600 prédicats) et sur une implantation en Prolog de CLP(FD) (130 prédicats utilisant fortement la métaprogrammation). Ces tests ont permis de documenter les prédicats définis par des annotations de types, et de vérifier la cohérence des programmes avec ces déclarations de type.

5.3 Simplexe

Participants : Jean-Claude Sogno [correspondant].

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).

L'intégration de ce solveur de contraintes dans GNU-Prolog est envisagée.

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

Participants : Philippe Codognet [correspondant].

Ce logiciel, réalisé par Yan Georget, est une extension du langage CLP(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 ISO Prolog

Participants : Pierre Deransart [correspondant].

Pierre Deransart, en collaboration avec AbdelAli Ed-Dbali, de l'Université d'Orléans, a développé 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.

Nous avons trouvé cette année une nouvelle traduction des programmes CC en logique linéaire qui permet de caractériser exactement l'ensemble des contraintes succès des calculs CC (non pas l'ensemble de leurs conséquences logiques), et qui lève toutes les restrictions concernant la caractérisation des suspensions en logique non-commutative. Ces résultats sont en cours de publication [26, 15].

D'autre part, une implémentation restreinte de la méthode de "phase model checking" présentée dans [3], a été réalisée par Sylvain Soliman en GNU-Prolog, en utilisant les contraintes arithmétiques pour la recherche d'espaces de phases permettant de prouver des propriétés de sureté des programmes LCC. Cette première implémentation prototype a permis de prouver automatiquement des propriétés de sureté de fonctionnement de programmes très simples de

protocoles de communication. Nous prévoyons de poursuivre ces travaux en collaboration avec Mitsuhiro Okada (Keio University, INRIA et LRI).

6.2 Implémentation de Prolog avec contraintes

Participants : Philippe Codognet, Daniel Diaz, Jean-Marie Geffroy.

Nous avons modifié le schéma de traduction de Prolog+contraintes utilisé par GNU-Prolog. Ce système est basé sur un schéma de compilation novateur et efficace reposant sur la définition d'un langage intermédiaire vers lequel est compilé Prolog. Ce langage est ensuite traduit vers le langage assembleur de la machine cible. En vue de faciliter l'écriture des traducteurs, ce langage est de bas niveau tout en restant indépendant de la machine et contient un jeu d'instructions réduit [13, 14].

Une fois compilé, un programme Prolog + contraintes donne lieu à un exécutable autonome (ne nécessitant pas la présence d'un émulateur à l'exécution). Ce schéma, initialement prévu pour des machines Unix 32 bits, a été généralisé pour prendre en compte les plateformes Windows ainsi que les processeurs 64 bits. De nouveaux traducteurs ont été écrits qui rendent aujourd'hui GNU-Prolog disponible sur les plateformes suivantes:

- ix86 / Linux
- ix86 / Win32
- ix86 / SCO
- ix86 / Solaris
- ix86 / FreeBSD
- ix86 / OpenBSD
- ix86 / NetBSD
- PowerPC / GNU/Linux
- sparc / SunOS
- sparc / Solaris
- alpha / linux
- alpha / OSF1
- mips / irix

6.3 Typage des programmes logiques avec contraintes

Participants : Emmanuel Coquery, Pierre Deransart, François Fages, Jan Smaus.

Les contraintes peuvent être vues comme un système de types dynamiques extrêmement puissant et l'on peut s'étonner que les langages de programmation par contraintes ne jouissent pas aujourd'hui de systèmes de types statiques, permettant de détecter des erreurs de programmation à la compilation, alors même que les domaines d'interprétation des contraintes fournissent un système de typage statique naturel.

Il est possible de typer les programmes logiques avec un système de types à la ML avec polymorphisme paramétrique mais cela ne permet pas de rendre compte des coercions 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 recherches. Notre approche est basée 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 de prédicats de métaprogrammation.

L'implémentation réalisée par Emmanuel Coquery cette année [24, 19] a permis de valider, sur un plan pratique, les hypothèses du système de types proposé par Fages et Paltrinieri pour les programmes logiques avec contraintes [23]. Notons que le système Wallace, développé par François Pottier dans le projet Cristal, est utilisé dans cette implémentation pour la résolution des inéquations de sous-typage.

La propriété de stabilité du typage par réduction ("subject reduction") est établie pour les programmes logiques avec contraintes dans un modèle d'exécution abstrait qui procède par accumulation de contraintes. L'extension de cette propriété aux opérations de résolution de contraintes a conduit à s'intéresser aux programmes logiques ayant un flot directionnel de données, défini par des modes. On montre que pour ces programmes, la propriété de stabilité est vérifiée dans le modèle d'exécution avec substitutions [17, 21].

Nous avons par ailleurs 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) [12, 20]. Ces conditions ont été obtenues en se plaçant à un plus haut niveau d'abstraction pour analyser la stabilité du typage par réduction. Dans cette approche on utilise la vue "grammaticale" de la programmation en logique, qui se base sur la notion d'arbre de preuve.

6.4 Contraintes linéaires diophantiennes

Participants : 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 equivalent, 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 clas-

sique (Hermite) fournissant une solution paramétrique avec moins de variables. Cette solution est reportée dans le sous-système original d'inéquations. Le nombre final de variables décroît du nombre d'équations éliminées, et dans certains cas, le calcul de la phase finale est immédiat. Cette stratégie a été appliquée en particulier pour des problèmes publiés dans la littérature, pour lesquels les temps de calcul sont passés de plusieurs minutes à quelques millisecondes. Ce travail est en cours de publication [28].

6.5 Contraintes flexibles

Participants : 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 avons travaillé à étendre dans cette théorie les techniques classiques de résolution de contraintes (mécanismes de cohérence locale et de de cohérence locale partielle) et développé de nouvelles techniques comme les mécanismes d'abstraction entre systèmes de contraintes. Le système CLP(FD,S) intègre l'implantation efficace de certaines de ces techniques.

6.6 Contraintes et recherche locale

Participants : Philippe Codognet, François Fages, 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 telles la conjonction de la cohérence d'arc suivie de l'énumération explicite des choix restants, la recherche locale est une heuristique incomplète qui ne fait une recherche exhaustive que dans un voisinage (local) d'une solution partielle. Ces idées sont en fait assez anciennes (certains algorithmes de résolution de problèmes du type "voyageur de commerce" étaient basées 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.

Dans [9] 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. 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. Des résultats préliminaires montrent que cette approche est prometteuse. Il reste cependant à intégrer dans un même cadre, pour de meilleurs résultats, techniques de recherche locale et techniques de réduction de domaines par propagation.

En collaboration avec Marc Shapiro, Microsoft Cambridge, François Fages étudie un problème combinatoire de réconciliation d'applications nomades, pour lequel il a développé deux prototypes: 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 locale, qui permet d'exploiter les solutions initiales et de résoudre le problème en ligne. Les travaux se poursuivent sur la méthode de recherche locale de façon à améliorer d'une part, la sortie des minima locaux, et d'autre part, la définition de grands voisinages par propagation de contraintes.

6.7 Contraintes 3D et réalité virtuelle

Participants : Frédéric Benhamou, Philippe Codognet, Yoann Fabre, Nadine Richard.

Le projet InViWo (*Intuitive Virtual Worlds*), qui fait l'objet de la thèse de Nadine Richard, a pour objectif de concevoir un outil intuitif de création de mondes virtuels, et plus particulièrement de proposer un langage de description de comportements d'agents vivants dans des environnements virtuels en 3D. Le projet se composera à terme de trois parties distinctes :

- une bibliothèque modulaire permettant de programmer et d'exécuter des comportements d'agents virtuels. L'exécution d'un monde virtuel doit tenir compte des éventuelles contraintes (locales ou globales) auxquelles peuvent être soumis les agents qui le composent.
- un langage déclaratif de haut niveau pour la manipulation de comportements d'agents, inspiré de TCC (Timed Concurrent Constraint) et intégrant un cadre générique de sélection de l'action permettant de coordonner divers comportements à exécuter.
- une interface graphique de création par composants d'agents et de mondes virtuels.

La bibliothèque de programmation et d'exécution des agents est en cours de réalisation dans l'environnement Java3D. Les modèles d'agent et de gestion des comportements implémentés permettent d'ores et déjà une conception de haut-niveau d'agents virtuels.

6.8 Environnements de mise au point

Participants : Pierre Deransart, Daniel Diaz, François Fages, Gérard Ferrand, Mohamed Amin Kaid, Jean-Claude Sogno.

Le livre des résultats du projet Esprit DiSCiPL ("Debuging Systems for Constraint Programming"), coordonné et co-édité par Pierre Deransart, est paru en septembre [2]. Les travaux sur ce thème se sont poursuivis d'une part par la préparation du projet RNTL OADymPPaC (voir section 8.1.1), et d'autre part par une étude spécifique pour la programmation linéaire en nombres entiers.

Afin de préparer le projet OADymPPaC, une étude préliminaire sur la réalisation de traces pour GNU-Prolog a été menée lors du stage de DEA de Mohamed Amin Kaid, en collaboration avec Mireille Ducassé (Projet Lande, INRIA/IRISA). Une sémantique opérationnelle à été définie, s'appuyant en partie sur les approches utilisées dans le cadre du projet DiSCiPl, qui a permis une extension au traitement des contraintes du modèle traditionnel ("boîtes de Byrd") utilisé en Prolog. Une implantation partielle a été réalisée avec une approche "interprète". Les résultats montrent la faisabilité d'une telle approche, mais des modèles plus généraux des solveurs de contraintes doivent pouvoir être utilisés afin de garantir une certaine généricité aux traces obtenues. L'objectif, dans le cadre du projet OADymPPaC, est que des outils interactifs d'analyse de comportement de programmes (et de résolution de contraintes) aussi variés que possibles puissent utiliser ces traces sans avoir à les redéfinir.

Le choix et l'implémentation d'une méthode de résolution de problèmes dans divers domaines de la programmation par contraintes sont en effet parfois malaisés. Pour un même type de problème, le choix d'une méthode, ou d'un paramétrage de méthode, peut s'avérer délicat, car dépendant de nombreuses caractéristiques du problème à résoudre. Au cours de la conception et de la réalisation de programmes d'optimisation en nombres entiers appliqués à la parallélisation de compilateurs, il était apparu indispensable de disposer d'outils permettant de controler souplement à la fois la validité et l'efficacité du code. Une méthode de visualisation d'algorithmes incrémentaux "tout-entier" est présentée dans [18]. Elle est basée sur la définition d'un "niveau de visualisation" dans un format immédiatement compréhensible (Latex) du comportement de l'algorithme, niveau pouvant à tout moment être modifié au cours du déroulement de l'algorithme interruptible. Cette possibilité permet de ne pas être submergé par de l'information non strictement utile et d'examiner à la loupe le comportement de l'algorithme à l'instant choisi. Cette stratégie est appliquée pour un travail en cours, sur l'étude et l'écriture de programmes de résolution de problèmes d'ordonnancement de tâches comportant des contraintes de disjonction.

6.9 Preuves de programmes par co-induction

Participants : Sorin Craciunescu, François Fages.

Un système de preuve par induction et co-induction est en cours de développement pour prouver la correction des programmes logiques avec contraintes. 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 bénéficie d'une collaboration avec Dale Miller et Jérémie Wajs de Penn State College University.

7 Contrats industriels (nationaux, européens et internationaux)

Nous sommes en contact avec le centre Microsoft Research de Cambridge pour l'élaboration d'un contrat de collaboration de recherche sur les problèmes combinatoires de réconciliation dans les applications nomades.

8 Actions régionales, nationales et internationales

8.1 Actions nationales

8.1.1 Projet OADymPPaC

Participants : Pierre Deransart, François Fages.

Le projet OADymPPaC (Outils pour l'Analyse Dynamique et la mise au Point de Programmes avec Contraintes), soumis en avril dans le cadre du RNTL, a été retenu et a débuté fin novembre [25].

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

Sans être la continuation proprement dite du projet DiSCiPl [2], il a pour ambition de surmonter quelques difficultés majeures rencontrées pour la réalisation d'environnements pour la PLC (portabilité et versatilité des outils de visualisation de l'espace des solutions).

La Programmation avec Contraintes est un ensemble de techniques de résolution de problèmes de grande complexité, autorisant un haut niveau d'expression et d'analyse des applications. Une grande partie des calculs détaillés est donc intégée dans des "solveurs". En contre partie il s'avère nécessaire de disposer d'outils d'"observation" de l'activité des solveurs, permettant d'analyser differents paramètres comme la complexité de l'espace de recherche, l'évolution des domaines des variables ou l'utilisation des contraintes à différents niveaux de granularité. De telles observations permettent de comprendre les processus de résolution et de concevoir les stratégies permettant de converger plus vite vers la solution recherchée.

Compte-tenu de l'état de l'art en matière de mise au point de programmes avec contraintes et de la variété des solveurs existants, le projet a pour objectif principal la définition de techniques de trace génériques et de formats d'échange afin de faciliter la définition d'outils d'observation et de mise au point. Ceci permettra, en parallèle, la définition et l'expérimentation de nouveaux outils avec deux champs d'application: la programmation avec contraintes et une meilleure compréhension du comportement des solveurs d'une part, et le développement de techniques génériques de visualisation d'information adaptées à l'analyse visuelle des traces produites par les phénomènes dynamiques d'autre part.

Les retombées attendues du projet sont de trois sortes: amélioration des compétences scientifiques et techniques des partenaires afin de maintenir l'avance technologique française dans ce domaine; amélioration des plateformes industrielles et académiques: solveurs de contraintes et leurs outils de mise au point pour Cosytec et modules de visualisation pour Ilog, logiciels libres comme GNU-Prolog, contribution à l'enseignement de la programmation avec contraintes.

Acteurs majeurs sur le marché de la programmation avec contraintes et de la visualisation, les partenaires industriels du projet ont pour but d'offrir une gamme complète d'outils dans leurs domaines respectifs. De ces outils et des résultats scientifiques du projet nous sommes en droit d'espérer une meilleure diffusion de ces techniques fondamentales.

8.1.2 Autres collaborations nationales

Des liens étroits existent avec le DI de l'ENS Ulm Paris (M. Fernandez, P. Cousot), 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'Université de Paris 6 et l'INRIA. Cette collaboration porte essentiellement sur des aspects langages et résolution de contraintes par méthodes d'intervalles sur les nombres réels. 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 progammes 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 obtenu dans le cadre du programme Galilée de coopération bilatérale avec l'Italie. Cette coopération concerne l'Université de Paris 6 (Philippe Codognet) et l'Université de Padoue (Francesca Rossi) sur le thème de la résolution de coUtraintes 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 European Network of Excellence in Computational Logic

Pierre Deransart était responsable du nœud de Rocquencourt «Programmation en Logique avec Contraintes» `http://www.compulog.org`. Ce réseau s'est terminé en Juin 2000.

8.2.7 Portugal

Pierre Deransart est responsable avec Marie-Christine Imbert (Inter) des relations avec le Portugal. Un appel à propositions INRIA/ICCTI pour des projets franco-portugais 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 juillet.

8.3.2 Brésil

Participants : Pierre Deransart.

Pierre Deransart est responsable avec Marie-Christine Imbert (Inter) 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 co-organisé une visite retour d'une délégation officielle du CNPq à l'INRIA (visite qui a eu lieu en novembre).

8.4 Visites, et invitations de chercheurs

Nous avons accueilli pour de courtes visites les chercheurs suivants: Pawel Rychlikowski et Tomasz Truderung (University of Wroclaw, Pologne), Rajeev Goré (Australian National University Canberra), Oltea Mihaela Herescu, Dale Miller, Catuscia Palamidessi et Jérémie Wajs (CSE, Penn State College, USA), Aviv Regev et Udi Shapiro (Weizmann Institute, Israël).

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

Philippe Codognet a été membre des comités de programmes de CP2000, 6th International Conference on Principles and Practice of Constraint Programming (Singapoure, septembre 2000), du stream "implementation and applications" de CL2000, First International Conference on Computational Logic (Londres, juillet 2000), de SAB2000, 6th International Conference on Adaptive Behavior (Paris, septembre 2000), de ISEA2000, 10th International Symposium on Electronic Arts (Paris, décembre 2000), et de JFPLC'2000. Il est président des comités de programme de ICLP2001, 16th International Conference on Logic Programming et de JFPLC'2001

Pierre Deransart a été membre des comités de programmes de PACLP'2000 de JFPLC'2000 et du premier "workshop" sur les environnements de programmation avec contraintes, associé à la conférence Constraint Programming CP'2000. Il est membre des comités de programme de JFPLC'2001 et PACLP'2001.

François Fages a été membre des comités de programme du Second Workshop on Rule-Based Constraint Reasoning and Programming, associé à la 6th International Conference on Principles and Practice of Constraint Programming CP'2000 (Singapour, septembre 2001), et des Neuvièmes Journées Francophones de Programmation Logique et Programmation par Contraintes, JFPLC'2000 (Marseille, juin 2000). Il est membre du comité de programme de JFPLC'2001 (Paris, avril 2001).

Gérard Ferrand est membre du comité de programme de JFPLC'2001, et a été membre du comite de programme du stream "Theory and Extension" de la conference CL'2000 (Computational Logic)

9.1.3 Organisation de manifestations

Philippe Codognet a co-organisé les workshops "Constraint Modeling and Solving", en conjonction avec ECAI2000, European Conference on Artificial Intelligence (Berlin, août 2000), et "Soft Constraints : Theory and Practice" en conjonction avec CP2000 (Singapour, septembre 2000). Philippe Codognet est membre du comité d'organisation de ISEA2000 (Paris, décembre 2000) et d'ASTI'2001 (Paris, avril 2001).

Pierre Deransart est Président de JFPLC'2001. A ce titre il est chargé de l'organisation de cette conférence. Il est également membre du comité d'organisation du congrès ASTI'2001.

François Fages a présidé cette année le comité de pilotage de la conférence ACM-Sigplan "Principle and Practice of Declarative Programming", PPDP. Il est 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 au grand public qui seront exposées à la Cité des Sciences à cette occasion.

9.1.4 Jurys

Philippe Codognet a été rapporteur des thèses de Stephane Bellot (Université d'Evry), de Frédéric Goualard (Université d'Orléans), de Stefano Bistarelli (Université de Pise, Italie) et d'Alain Lioret (Université de Paris 8). Il a été président des jurys de Guillaume Hutzler (Université de Paris 6) et d'Anne Liret (Université de Paris 6).

Pierre Deransart a été raporteur de la thèse d' Erwan Jahier (INSA-IRISA, Rennes).

François Fages a été rapporteur de la thèse de doctorat de Pawel Rychlikowski et Tomasz Truderung (Université de Wroclaw, Pologne), et a participé au jury d'habilitation de Maribel Fernandez (Université de Paris Sud). Il est rapporteur de la thèse de Thi-Bich-Hahn (Université de Provence, Marseille). Il est 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.

Gérard Ferrand est rapporteur de la thèse de doctorat de Sylvain Merchez (CRIL, Lens) et membre du jury d'HDR de Lakhdar Sais (CRIL, Lens).

9.1.5 Vulgarisation scientifique

Emmanuel Coquery a animé une session sur "Internet" au Bar des Sciences qui s'est tenu en novembre à Versailles.

Sylvain Soliman a donné une conférence sur ""La force naît des contraintes", L. De Vinci" au Colloquium Junior de l'INRIA Rocquencourt.

9.1.6 Responsabilités associatives

Philippe Codognet est membre du comité executif 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),

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

9.2 Enseignement

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.

10 Bibliographie

Livres et monographies

[1] P. Codognet, F. Pachet (éditeurs), *Special issue on Constraints for multimedia applications, CONSTRAINTS, an International Journal*, Kluwer Academic Press, Decembre 2000.

[2] P. Deransart, J. Małuszyñski, M. Hermenegildo (éditeurs), *Analysis and Visualization Tools for Constraint Programming, LNCS, 1870*, Springer Verlag, Septembre 2000.

Articles et chapitres de livre

[3] F. Fages, P. Ruet, S. Soliman, «Linear concurrent constraint programming: operational and phase semantics», *Information and Computation*, 2000, à paraître.

Communications à des congrès, colloques, etc.

[4] J. Arsouze, G. Ferrand, A. Lallouet, «A CSP view of CLP», *in : Proc. Workshop Fixed Points in Computer Science FICS'2000*, Paris, juillet 2000.

[5] J. Arsouze, G. Ferrand, A. Lallouet, «Une semantique co-inductive pour la propagation de contraintes et le labeling», *in : Programmation en logique avec contraintes, actes des JFPLC'2000*, Touraïvane (éditeur), Hermès, Juin 2000.

[6] S. Bistarelli, P. Codognet, Y. Georget, F. Rossi, «Abstracting Soft Constraints», *in : ERCIM/CompulogNet Working Group on Constraints, LNAI 1865*, 1865, Springer-Verlag, 2000.

[7] S. Bistarelli, P. Codognet, Y. Georget, F. Rossi, «Labeling and Partial Local Consistency for Soft Constraint Programming», *in : Proceedings of PADL'00, 2nd international workshop on Practical Aspects of Declarative Languages, LNCS 1753*, Springer-Verlag, Boston, USA, Janvier 2000.

[8] S. Bistarelli, P. Codognet, F. Rossi, «An Abstraction Framework for Soft Constraints, And Its Relationship with Constraint Propagation», *in : proceedings of SARA 2000, 4th International Symposium on Abstraction, Reformulation, and Approximation, LNAI 1864*, Springer-Verlag, Austin, Texas, USA, Juillet 2000.

[9] P. CODOGNET, «Adaptive Search : Preliminary Results», *in : proceedings ERCIM / CompulogNet Workshop on Constraints*, Venise, Italie, Juin 2000.

[10] P. CODOGNET, «Animating virtual creatures by constraint-based adaptive search», *in : Proceedings of VSMM2000, 6th International Conference onVirtual Systems and Multimedia*, IOS Press, Gifu, Japon, Octobre 2000.

[11] P. CODOGNET, «A Constraint-based Language for Virtual Agents», *in : ERCIM/CompulogNet Working Group on Constraints*, *LNAI 1865*, 1865, Springer-Verlag, 2000.

[12] P. DERANSART, J.-G. SMAUS, «Les programmes bien typés ont tout bon», *in : Programmation en logique avec contraintes, JFPLC'2000*, Touraïvane (éditeur), Hermès, p. 49–66, Juin 2000.

[13] D. DIAZ, P. CODOGNET, «GNU Prolog: beyond compiling Prolog to C», *in : Proceedings of PADL'00, 2nd international workshop on Practical Aspects of Declarative Languages*, *LNCS 1753*, Springer-Verlag, Boston, USA, Janvier 2000.

[14] D. DIAZ, P. CODOGNET, «The GNU Prolog System and its Implementation», *in : Proceedings of SAC'00, 14th ACM Symposium on Applied Computing*, ACM Press, Come, Italie, Avril 2000.

[15] F. FAGES, «Concurrent constraint programming and linear logic (invited talk)», *in : Proceedings of ACM Sigplan conference on Principles and Practice of Declarative Programming PPDP'2000*, ACM Publ., Montreal, Canada, September 2000.

[16] G. FERRAND, W. LESAINT, A. TESSIER, «Value Withdrawal Explanation in CSP», *in : Proc. Fourth International Workshop on Automated Debugging AADEBUG'2000*, Münich, août 2000.

[17] J.-G. SMAUS, F. FAGES, P. DERANSART, «Using Modes to Ensure Subject Reduction for Typed Logic Programs with Subtyping», *in : Proceedings of FSTTCS '2000*, *LNCS*, 1974, Springer-Verlag, 2000.

[18] J.-C. SOGNO, «Visualization of Incremental All-Integer Algorithms», *in : Proceedings of CP2000 Workshop: Analysis and Visualisation of Constraint Programs and Solvers*, Singapore, September 22 2000.

Rapports de recherche et publications internes

[19] E. COQUERY, «Implantation d'un système de types prescriptifs pour la programmation logique avec contraintes», *Rapport de dea, Université Paris XI Orsay*, INRIA, 2000.

[20] P. DERANSART, J.-G. SMAUS, «Typed logic programs are not wrong», *Research report*, INRIA, 2000, version longue du papier JFPLC'2000, à paraître.

[21] J.-G. SMAUS, F. FAGES, P. DERANSART, «Using Modes to Ensure Subject Reduction for Typed Logic Programs with Subtyping», *rapport de recherche*, INRIA RR-4020, Octobre 2000, Version complète de l'article présenté à FSTTCS'2000, `http://www.inria.fr/rrrt/rr-4020.html`.

Divers

[22] P. CODOGNET, F. ROSSI, «Programming and solving with soft constraints», 2000, Tutorial notes, presented at AAAI'2000 and ECAI'2000.

[23] E. COQUERY, F. FAGES, « Typed Constraint Logic Programs: checking domain coercions and metaprograms through subtyping», octobre 2000, Research report.

[24] E. COQUERY, «`Tclp`: a generic type checker for constraint logic programs», octobre 2000.

[25] P. DERANSART, « OADymPPaC», juillet 2000, Annexe technique.

[26] F. FAGES, S. SOLIMAN, « Advances in the linear logic semantics of concurrent constraint languages», décembre 2000, Research report.

[27] M. A. KAID, «Etude et réalisation de trace en GNU-Prolog», septembre 2000, Rapport de DEA.

[28] J.-C. SOGNO, « The Shrinking Strategy for Solving Linear Diophantine Systems», mai 2000, Research report.