INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Projet Contraintes

Programmation par Contraintes

Rocquencourt

THÈME 2A

Table des matières

1. Composition de l'équipe

Contraintes est un projet commun avec le LIP6 de l'Université Pierre et Marie Curie Paris 6.

Responsables scientifiques
François Fages [DR, INRIA]
Philippe Codognet [Professeur, Université de Paris 6]

Responsable permanent
Pierre Deransart [DR, INRIA]

Assistante de projet
Josy Baron [AJT, INRIA]

Personnel Inria
Jean-Claude Sogno [DR, jusquau 30/4/2002]
Sylvain Soliman [CR]

Collaborateurs extérieurs
Frédéric Benhamou [Professeur, Université de Nantes]
Daniel Diaz [Maître de conférence, Université de Paris 1]
Gérard Ferrand [Professeur, Université d'Orléans]

Ingénieur expert
Guillaume Arnaud

Doctorants
Nathalie Chabrier [Bourse INRIA, depuis le 1/11/2002]
Emmanuel Coquery [allocataire MESR, INRIA]
Sorin Craciunescu [allocataire AMX jusqu'au 1/9/2002, ATER depuis]
Yoann Fabre [allocataire MESR, Paris 6]
Luc Hernandez [allocataire MESR, Paris 6]
Ludovic Langevine [Bourse INRIA région]

Stagiaires
Anupam Agarwal [du 1/5/2002 au 30/7/2002]
Nathalie Chabrier [du 1/4/2002 au 30/9/2002]
Rémy Hæmmerlé [du 1/4/2002 au 30/9/2002]
Rachid Kaddouche [du 1/4/2002 au 30/6/2002]
Salim Kalla [du 1/4/2002 au 30/9/2002]
Akash Lal [du 1/5/2002 au 30/7/2002]

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 extensi ons 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 trois domaines applicatifs :

- la résolution de problèmes combinatoires, domaine d'excellence de la programmation par contraintes aujourd'hui ;
- 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 ;
- et la bioinformatique, où, au delà du traitement de problèmes combinatoires en biologie, les langages concurrents avec contraintes offrent des outils puissants de modélisation et d'analyse des processus biologiques multiéchelles.

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. 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'application

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 travail 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.

4.3. Bioinformatique

Mots clés : *biologie des systèmes*, *processus biologiques*, *réseaux de régulation entre gènes*, *réseaux d'intéraction biochimiques*, *calculs de processus*, *« model checking » symbolique*.

La biologie s'est clairement engagée ces dernières années dans un travail d'élucidation des processus biologiques de haut niveau en termes de leurs bases biochimiques à l'échelle moléculaire. Avec la fin des années 90, le front de la recherche en bioinformatique a évolué ; passant de l'analyse de la séquence génomique à l'analyse de données diverses produites en masse par les technologies dites « post-génomiques » (expression des ARN et des protéines, SNP et haplotypes, interactions protéine-protéine, structures 3D, etc.). Cet effort de « désassemblage » par identification et mesure de certaines caractéristiques des constituants élémentaires (gènes et protéines) commence à pouvoir servir de base à l'effort systématique inverse : la reconstitution des mécanismes biologiques au sein desquels ces constituants exhibent une fonction.

La production en masse de données post génomiques, telle que l'expression des ARNs, la synthèse de protéines, et les interactions protéines-protéines, soulève le besoin d'un effort parallèle important sur la représentation formelle des *processus* biologiques. La conception d'outils formels pour modéliser les processus biomoléculaires, et pour raisonner sur leur dynamique apparaît comme une voie de recherche obligatoire à laquelle les techniques de vérification formelle, et de programmation concurrente avec contraintes en particulier, peuvent contribuer énormément.

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. A ce jour plus de 50000 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. CLPGUI

Participant : François Fages [correspondant].

CLPGUI est une interface graphique générique, écrite en Java, qui permet de visualiser et piloter l'exécution d'un programme logique avec contraintes. CLPGUI est actuellement disponible pour GNU-Prolog et Sicstus-Prolog. CLPGUI a été développé à la fois à des fins d'enseignement et pour la mise au point de programmes complexes sur des données réelles.

L'interface graphique est composée de plusieurs fenêtres : une console principale et plusieurs visualiseurs dynamiques, 2D et 3D, de l'arbre de recherche et des domaines des variables.

Avec CLPGUI il est possible d'exécuter de façon incrémentale n'importe quel but ou contrainte, retourner en arrière, et recalculer un état quelconque représenté par un nœud dans l'arbre de recherche. Le niveau de granularité de l'arbre de recherche est défini par des annotations dans le programme CLP/CC.

L'implémentation réalisée est diffusée librement sous la licence LGPLC à l'URL http://contraintes.inria.fr/~fages/CLPGUI/. Une version de ce logiciel adapté aux problèmes de placements 2D a remporté le prix du « Best User Interface Concept » au concours du 13ième « ACM Symposium on User Interface Software and Technology » UIST'2002.

CLPGUI sera étendu pour gérer les traces génériques d'exécution et de propagation des contraintes qui sont en cours de définition dans lecadre du projet RNTL OADymPPac.

5.3. 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. 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. Ce système de types a été étendu pour intégrer la surcharge de type. La surcharge de type permet de donner deux types différents à un même symbole de fonction ou de prédicat et permet ainsi de pouvoir utiliser de plusieurs manières un tel symbole, pratique répandue en Prolog.

L'implémentation réalisée ainsi qu'une version de démonstration en ligne sont disponibles à l'URL http://contraintes.inria.fr/~coquery/tclp. TCLP a été réécrit en Prolog, avec l'utilisation des *Constraint Handling Rules* (CHR) pour la résolution des contraintes de sous-typage et la gestion de la surcharge. Le système TCLP a été testé sur les prédicats prédéfinis de ISO-Prolog, 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).

5.4. Simplexe

Participant : 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).

Par ailleurs, une implantation du Simplexe en nombres entiers développée par Jean-Claude Sogno est également disponible. Ce logiciel est utilisé par François Irigoin à l'Ecole des Mines pour la compilation de programmes sur machines parallèles.

5.5. 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://contraintes.inria.fr/~deransar/prolog/.

6. Résultats nouveaux

6.1. Langages concurrents avec contraintes et logique linéaire

Participants : François Fages, Sylvain Soliman.

L'implémentation de la méthode de vérification automatique par « Phase Model Checking » a été poursuivie par Sylvain Soliman en GNU-Prolog [31]. 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 permet de prouver automatiquement des propriétés de sûreté de fonctionnement de programmes simples de protocoles de communication. Les travaux se poursuivent 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, François Fages.

Cette année a vu l'introduction de la surcharge de types pour les symboles de fonction et de prédicats dans le système de types de TPLP [6][7]. L'ajout de la surcharge permet de gérer différentes utilisations d'un même symbole, comme par exemple, l'opérateur binaire '-' qui est utilisé en Prolog pour les opérations arithmétiques et pour le codage des paires.

L'implantation de la surcharge utilise le principe Andorra, qui permet de repousser l'exécution des points de choix après celle des parties déterministes. Cette stratégie, ainsi que la simplification de la structure des types qui résulte de la surcharge au lieu du sous-typage, permet d'obtenir des temps de typage avec surcharge comparables à ceux sans surcharge, contre une grande flexibilité d'utilisation.

Parallèlement, nous cherchons à relâcher l'hypothèse de treillis imposée à la structure des types. Plus précisément, nous travaillons actuellement sur la décidabilité des contraintes de sous typage non-structurel dans les structures de quasi-treillis.

6.3. GNU-Prolog

Participants : Philippe Codognet, Daniel Diaz.

En collaboration avec Bart Demoen et Ruben Vandeginste (Université Catholique de Louvain) nous avons implanté un ramasse-miettes pour GNU Prolog. Celui-ci est encore en version de test mais son implantation dans la version officielle de GNU Prolog est prévue courant 2003.

Dans le cadre de la collaboration franco-portugaise (INRIA-ICCTI) avec Salvador Abreu (Université de Evora) nous avons étudié une alternative aux modules Prolog classiques : la programmation contextuelle. L'idée étant de maintenir une pile de contextes appelants et d'utiliser cette pile pour trouver quel prédicat doit être invoqué. Cette approche est très flexible et permet d'implanter beaucoup d'autres systèmes de modules (classiques, orientés-objets,...). Son unique inconvénient est d'engendrer des appels indirects (devant être résolus dynamiquement à l'exécution plutôt que statiquement à la compilation). Heureusement il existe bon nombre de cas où les appels les plus coûteux peuvent être optimisés. Le prototype réalisé en GNU Prolog offre déjà des performances prometteuses.

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 [1]. 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, Rachid Kaddouche.

Cette année Rémi Hæmmerlé a introduit un mécanisme général de co-routines et de variables attribuées dans GNU-Prolog, qui permet de définir des modes d'exécution concurrents de buts qui s'exécutent au moment de l'unification de variables [22]. Ce mécanisme général a servi de premier prototype de l'implémentation du noyau LCC. Il a servi d'autre part à étendre GNU-Prolog pour traiter les contraintes linéaires sur les nombres réels suivant le schéma CLP($\mathcal{R}$) en réalisant une interface avec une implémentation, rendue incrémentale, de l'algorithme du Simplexe LP-Solve. Cette interface permet de traiter les contraintes linéaires sur les réels conjointement aux contraintes sur les domaines finis de GNU-Prolog.

GNU-Prolog a également été étendu, à la suite des travaux d'Alexis Saurin, par une interface permettant de définir en langage C des contraintes globales [23]. La contrainte `fd_all_different(List)` utilisant l'algorithme de Régin (couplage maximal dans un graphe biparti) a été implémentée de cette façon dans une version prototype de GNU-Prolog.

6.6. 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, nommée Recherche Adaptative. 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 [5]. 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 en cours de développement.

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.

6.7. Réconciliation

Participants : François Fages, Akash Lal.

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. Nous avons montré que le problème central responsable de la complexité NP de la reconicliation est un problème de recherche dans un graphe du plus grand sous-graphe sans cycle. Les travaux se sont poursuivis sur la définition d'une contraintes globale pour les problèmes de plus grand sous-graphe sans cycle dans un graphe [25] qui a permis d'améliorer les performances du prototype de reconcilation réalisé en programmation logique avec contraintes.

6.8. Contraintes 3D et Réalité Virtuelle

Participants : Philippe Codognet, 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. 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 [4]. 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. Elle a été optimisée et interfacée avec divers moteurs 3D.

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

6.9. « Model checking » symbolique en biologie des réseaux moléculaires

Participant : Nathalie Chabrier, François Fages, Sylvain Soliman.

Dans le cadre de l'ARC CPBIO nous avons exploré l'utilisation de techniques de « model checking » pour interroger en logique temporelle CTL des modèles de réseaux biochimiques. Notre démarche repose sur l'utilisation des techniques de « model checking » symbolique pour l'interrogation des modèles purement logiques (qualititatifs), et de techniques de « model checking » avec contraintes pour les modèles quantitatifs décrits par des systèmes différentiels [20][14][3][19].

Nous avons modélisé par un système de transition concurrent de 732 règles et 532 variables (portant sur 165 protéines de base) le contrôle du cycle cellulaire chez les mammifères, d'après la carte d'interaction biomoléculaire de Kohn suivant la modélisation en π-calcul de Marc Chiaverini et Vincent Danos développée dans l'ARC CPBIO. Nous avons montré que les questions d'accessibilité (synthèse de protéines), de chemins (point de restriction, points de contrôle), et d'états stables s'expriment simplement en logique CTL, et que l'outil de « model checking » symbolique NuSMV permet d'interroger automatiquement le modèle avec des temps de réponse de quelques secondes CPU.

Nous avons également montré la faisabilité de cette démarche pour interroger des modèles quantitatifs de régulation entre gènes. La méthode consiste à discrétiser le temps comme dans les méthodes classiques d'Euler ou Runge-Kutta, et à appliquer des techniques de « model checking » avec contraintes (en l'occurrence l'outil de programmation logique avec contraintes DMC de Podelski et Deganno) pour interroger de façon symbolique les propriétés temporelles quantitatives du système.

Ce travail ouvre de nombreuses perspectives, tant du point de vue de la biologie des systèmes (une collaboration est en cours de démarrage avec un laboratoire de cancérologie à Nancy), que du point de vue des méthodes de « model checking » avec contraintes (liens avec les automates hybrides et les outils de vérification hybrides, combinaison avec des procédures de recherche d'états,...).

6.10. Environnements de mise au point

Participants : Anupam Agarwal, Guillaume Arnaud, Pierre Deransart, Daniel Diaz, François Fages, Gérard Ferrand, Salim Kalla, Ludovic Langevine.

L'objectif général est de permettre l'utilisation de différents outils de débogage sur différentes plateformes de solvers domaines finis. L'approche repose donc sur la définition d'une trace générique à partir d'un modèle général de solveurs domaines finis. Cette trace est implantée sur différents solveurs dans le cadre du projet OADymPPaC (cf 8.1.2), notre contribution portant en particulier sur l'implantation d'un traceur pour GNU-prolog. Nous avons également porté nos efforts sur l'analyse du débogage et la définition d'outils adaptés. Notre contribution a porté en particulier sur les points suivants : l'extention du modèle de trace générique au contrôle, une première implantation en GNU-prolog, une étude du débogage centrée sur les problèmes riches en symétries, et l'étude des interactions avec la réalisation de l'outil de débogage CLP-GUI.

6.10.1. Extention du modèle de trace générique

Un modèle complet a été présenté dans [18], ainsi qu'une analyse de son adéquation avec plusieurs solveurs (Ilog, Choco, GNU-Prolog). Afin de tester la robustesse du modèle générique nous avons cherché également à utiliser ce modèle pour tracer des programmes utilisant le solveur domaine finis de CHR (Constraint Handling Rules) dont l'approche est basée sur des règles de réécriture et donc très différente. Un premier résultat [24] montre la possibilité de spécifier et implanter notre modèle générique dans CHR(FD). L'implantation partielle qui a été réalisée est basée sur un métainterprète.

6.10.2. Implantation d'un traceur en GNU-Prolog

L'étape suivante a donc consisté à entreprendre l'implantation d'un traceur pour GNU-Prolog. Une première version devrait être opérationnelle d'ici la fin de l'année. Il permettra alors, grâce au format de trace basé sur le standard XML défini dans le projet OADymPPaC (cf. 8.1.2), d'entreprendre une évaluation de plusieurs outils de débogage.

6.10.3. Etude du débogage des problèmes avec symétries

Les études générales sur l'état de l'art du débogage de solveurs de contraintes achevées [26] [28], nous nous sommes concentrés sur l'analyse de propriétés remarquables de certains problèmes : les symétries. Leur intérêt est leur fréquence, leur impacte sur l'efficacité de l'obtention de solutions et la difficulté qu'il peut y avoir à les prévoir pour des problèmes ordinaires.

Nous avons donc entrepris une étude des propriétés de certaines visualisations pour des problèmes comportant des symétries. L'objectif à terme est de pouvoir proposer des stratégies d'élimination des symétries détectées par la seule observation de ces visualisations particulières de la trace (il s'agit essentiellement des symétries non connues au départ). Cette étude nécessite pour pouvoir être menée à bien de disposer du traceur. En attendant, une implantation de l'élimination totale par la méthode SBDS des symétries connues a été réalisée pour GNU-prolog. Ceci nous permettra ainsi d'évaluer l'efficacité des stratégies ainsi améliorées.

6.10.4. Débogage et interaction

La trace générique est potentiellement volumineuse et risque d'être impraticable. C'est dans ce but qu'une approche « à la opium » du traceur à été entreprise. Tout outil de débogage adresse des « requêtes » au traceur qui de ce fait n'envoie (donc ne trace) que les éléments utiles à l'outil. Le démarche, adaptée de l'approche « opium » pour Prolog, est décrite dans [10][11]. Avec cette approche nous avons aisément produit des traces (à l'aide du méta-solveur élaboré l'année dernière) pour différents types de visualiseurs (VRML, Discovery de Ilog, visualiseurs proposés par l'EMN cf.8.1.2).

Développé en parallèle, l'interface graphique CLPGUI [13] a été perfectionné. Au delà de la visualisation 3D, il permet d'interagir avec le programme CLP en ajoutant incrémentalement de nouvelles contraintes ou de nouveaux buts, et en pilotant les retours en arrière. Dés que le traceur GNU-Prolog sera opérationnel, il sera possible d'en faire une version basée sur la trace générique. Cet outil nous permet en particulier d'étudier certaines interactions.

6.11. Preuves de programmes par co-induction

Participant : Sorin Craciunescu.

Les methodes formelles de preuve de programmes logiques sont une autre direction de recherche du projet. L'équivalence des programmes logiques est étudiée par Sorin Craciunescu. Dans [8], un système de preuve formel qui permet de démontrer l'équivalence de programmes logiques est présenté. Le langage traité est le langage CLP étendu avec le quantificateur universel. Deux programmes CLP$\forall$ sont équivalents si l'ensemble des succès est le même pour les deux programmes. Le systeme de preuve est basé sur la logique classique du premier ordre à laquelle est ajoutée une règle d'induction. Cela permet de prouver l'équivalence des succès finis. En remplaçant la règle d'induction par une règle de coinduction, le système permet aussi de prouver l'équivalence des succès infinis ce qui est utile pour modéliser les systèmes réactifs. Un verificateur de preuve pour ce formalisme a été implanté en Prolog et un assistant intéractif de preuve est en phase finale de réalisation.

7. Contrats industriels

7.1. COSYTEC

Participant : François Fages.

Notre collaboration avec Cosytec s'effectue principalement dans le cadre du projet RNTL OaDymPpac (voir plus bas). On peut citer la définition en commun du format de trace générique des solveurs de contraintes sur les domaines finis, le transfert technologique du format de trace générique dans l'outil CHIP, et le transfert de technologie du logiciel CLPGUI pour la visualisation dynamique de l'exécution des programmes CHIP.

7.2. ILOG

Participant : François Fages.

Notre collaboration avec ILOG s'est principalement effectuée dans le cadre du projet RNTL OaDymPpac. Elle a notamment porté sur l'évaluation du logiciel Discovery d'ILOG pour la visualisation des traces d'exécution dans le cadre de la programmation avec contraintes.

Une nouvelle collaboration est en cours de démarrage avec la labélisation du projet RNTL MANIFICO sur la metacompilation non-intrusive du filtrage avec contraintes dans les langages de règles.

7.3. Microsoft

Participant : François Fages.

Le contrat de collaboration de recherche avec le centre Microsoft Research de Cambridge (équipe de Marc Shapiro) s'est poursuivi 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. Action de Recherche Coopérative CPBIO

Participants : François Fages, Nathalie Chabrier, Sylvain Soliman.

Nous avons lancé cette année une Action de Recherche Coopérative (ARC) intitulée « Calculs de Processus et Biologie des Réseaux Moléculaires » avec Alexander Bockmayr, ModBio, LORIA, Nancy, Vincent

Danos, CNRS, PPS, Paris 7, Magali Roux Rouquié, Institut Pasteur, et Vincent Schächter, Genoscope Evry (précédemment Hybrigenics). Le thème central est la recherche de langages formels de modélisation et d'outils d'analyse automatique des intéractions biomoléculaires à l'échelle de la cellule. Les principaux exemples traités sont le contrôle du cycle cellulaire chez les mammifères, les réseaux de régulation entre gènes et l'épissage alternatif.

8.1.2. Projet OADymPPaC

Participants : Anupam Agarwal, Guillaume Arnaud, Pierre Deransart, Daniel Diaz, François Fages, Gérard Ferrand, Salim Kalla, Ludovic Langevine.

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).

L'essentiel du travail scientifique concernant notre équipe a été décrit dans la section 6.10. Pierre Deransart assure de plus le pilotage du projet et son administration. A ce titre les actions suivantes ont été poursuivies.

- Animation du site WEB du projet http://contraintes.inria.fr/OADymPPaC. 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 première revue (21-22 février 2002) à l'INRIA-Rocquencourt [30].
- Lancement et co-organisation de plusieurs réunions thématiques sur les interactions (28 février), organisation à Nice lors des JFPLC (28 mai) et à l'INRIA (22 août), sur la réalisation de traceur à Rennes (28 octobre) et une réunion générale (15 octobre) de coordination et d'information.
- Edition d'un poster et d'une *fact sheet* de présentation du projet, présentés lors des journées RNTL (23-24 octobre).
- Présentation du projet OADymPPaC à l'occasion des journées RNTL du SITEF à Toulouse (23 et 24 octobre) et du *Chip-users club* (29 novembre 2002).

8.1.3. Autres collaborations nationales

Des liens étroits existent avec l'IRIN de Nantes (F. Benhamou, F. Goualard, L. Granvillier), le laboratoire PPS de Paris (V. Danos), le LIFO d'Orléans (G. Ferrand, A. Ed-Dbali, A. Lallouet, A. Tessier), le projet Landes de l'IRISA (M. Ducassé), le projet Protheo de l'INRIA (C. et H. Kirchner, P.E. Moreau), le projet ModBIO du LORIA (A. Bockmayr, A. Courtois, D. Eveillard).

8.2. Actions européennes

8.2.1. TMR Linear Logic in Computer Science

Participants : François Fages, Sylvain Soliman.

Nous avons participé 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). Ce réseau se termine cette année.

8.2.2. ERCIM working group on Constraints

François Fages est le nouveau président du groupe de travail ERCIM sur les contraintes, et Sylvain Soliman le nouveau secrétaire de ce groupe. Ce réseau regroupe de 15 équipes européennes, voir http://contraintes.inria.fr/~soliman/ercim.

8.2.3. Réseau d'excellence COLOGNET

Participants : Pierre Deransart, Daniel Diaz, François Fages, Sylvain Soliman.

Nous sommes membre de ce réseau d'excellence et appartenons au groupe sur la programmation logique avec contraintes CLP.

8.2.4. 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.

8.2.5. 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.

8.2.6. Portugal

Pierre Deransart est responsable avec Martine Lecorre (DRI) des relations bilatérales avec le Portugal. Un appel à propositions INRIA/ICCTI pour des projets franco-portugais (juillet 2002) ainsi que leur sélection ont été réalisés.

8.2.7. Autres collaborations européennes

Des liens étroits existent avec le Max Planck Institute, Saarbrück, Allemagne (A. Podelski), L'université de Padoue, Padova, Italie (F. Rossi), L'université de Linköping, Suede (J. Maluzinski), l'université Ludwig Maximilian, Münich, Allemagne (T. Fruehwirth, S. Abdennadher).

8.3. Actions internationales

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

Participants : Sorin Craciunescu, François Fages, Sylvain Soliman.

Le contrat de collaboration NSF-CNRS 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) s'est achevé cette année.

8.3.2. Brésil

Participant : Pierre Deransart.

Pierre Deransart est responsable avec Marie-Christine Imbert (DRI) des relations bilatérales avec le Brésil. Cette année à été marquée par la préparation d'accords avec les fondations pour la recherche de l'état de São Paulo (FAPESP) et de l'état de Rio de Janeiro (FAPERJ), ainsi que pour la mise en place de la participation de l'école polytechnique de la USP au programme « internship ».

8.4. Visites, et invitations de chercheurs

Nous avons accueilli pour de courtes visites les chercheurs suivants : Rafaël Ramirez-Melendez (post doc ERCIM embauché à l'université de Barcelone).

9. Diffusion des 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

Pierre Deransart a été membre des comités de programme de FLOPS'2002 (*Sixth International Symposium on Functional and Logic Programming*) et des JFPLC'2002 (Nice, juin 2002). Il est membre du comité de programme de JFPLC'2003 (Amiens, juin 2003).

François Fages a été membre du comité de programme de ICLP'2002, *17th International Conference on Logic Programming* (Copenhague, juillet 2002), PPDP'2002, *Principles and Practice of Declarative Programming* (Pittsburgh, septembre 2002), workshop associé RULE'2002, WFLP'2002, *Workshop on Functional and Logic Programming* (Italie, mai 2002), joint Ercim Constraints Colognet CLP meeting (Cork, Irelande, juin 2002), et du workshop RNTL (Toulouse, octobre 2002). Il est membre du comité de programme de CMSB'2003 *Computational Methods in Systems Biology* (Rovereto, Italie, février 2003) et des JFPLC'2003 (Amiens, juin 2003).

9.1.3. Organisation de manifestations

9.1.4. Jurys

François Fages a été rapporteur de la thèse d'habilitation à diriger des recherches d'Eric Monfroy (Université de Nantes), de la thèse de doctorat de Fabrice Parennes (Université Paris 6), et de la thèse de doctorat de Jean-Vincent Loddo (Université Paris 7). Il a participé au jury de la thèse de Wendelin Serwe (INPG Grenoble). Il a été membre du jury d'admission du concours CR2 de l'INRIA Rocquencourt, du conseil scientifique de l'Institut Liapunov, des commissions de spécialistes des Universités Denis Diderot, Paris 7, et Paris Sud, Paris 11, et du comité scientifique de l'ONERA.

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. Il est membre du comité de pilotage de la conférence ACM-Sigplan *Principles and Practice of Declarative Programming*, PPDP.

François Fages est président de l'AFPLC, président du groupe de travail ERCIM sur les contraintes et membre du comité de pilotage de la conférence ACM-Sigplan *Principles and Practice of Declarative Programming*, PPDP.

Sylvain Soliman est secrétaire du groupe de travail ERCIM sur les contraintes.

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 ont donné 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. Ecole Jeunes Chercheurs, Rennes

François Fages a donné un cours sur la Programmation Logique avec Contraintes à l'Ecole Jeunes Chercheurs du GDR ALP du CNRS.

10. Bibliographie

Articles et chapitres de livre

[1] S. BISTARELLI, P. CODOGNET, F. ROSSI. *Abstraction for Soft Constraints : Framework, Properties, Examples.* in « Artificial Intelligence », numéro 2, volume 139, 2002.

Communications à des congrès, colloques, etc.

[2] J. ARSOUZE, G. FERRAND, A. LALLOUET. *Arbres d'itérations chaotiques pour décrire la résolution des CSPs.* in « Journées Francophones de Programmation en Logique et par Contraintes », Hermès, éditeurs M. RUEHER., Nice, France, 2002.

[3] N. CHABRIER, F. FAGES. *Symbolic model checking of biological systems.* in « Poster Abstracts of the European Conference on Computational Biology, ECCB'02 », éditeurs H. L. T. LENGAUER, R. CHRISTMANN., Saarbrücken, Germany, Octobre, 2002.

[4] P. CODOGNET. *Multi-Goal Pathfinding for Autonomous Creatures in Virtual Words.* in « Entertainment Computing : Technologies and Applications », Kluwer Academic Publisher, éditeurs K. NAKATSU, J. HOSHINO., 2002.

[5] P. CODOGNET, D. DIAZ, C. TRUCHET. *The Adaptive Search Method for Constraint Solving and its Application to Musical CSPs.* in « First International Workshop on Heuristics », éditeurs J.-K. HAO., Beijing, Chine, 2002.

[6] E. COQUERY, F. FAGES. *Surcharge et sous-typage dans TCLP.* in « Journées Francophones de Programmation en Logique et par Contraintes », Hermès, éditeurs M. RUEHER., pages 273-287, Nice, France, Juin, 2002.

[7] E. COQUERY, F. FAGES. *TCLP : overloading, subtyping and parametric polymorphism made practical for constraint logic programming (abstract).* in « International Conference on Logic Programming, ICLP'2002 », série Lecture Notes in Computer Science, volume 2401, Springer-Verlag, pages 480, Copenhagen, Danmark, 2002.

[8] S. CRACIUNESCU. *Proving the Equivalence of CLP Programs.* in « International Conference on Logic Programming, ICLP'2002 », série Lecture Notes in Computer Science, volume 2401, Springer-Verlag, pages 287-301, Copenhagen, Danmark, 2002.

[9] P. DERANSART, M. DUCASSÉ, L. LANGEVINE. *A Generic Trace Model for Finite Domain Solvers.* in « Proceedings of UICS'2002, workshop on User Interaction in Constraint Satisfaction, associated to Constraint Programming CP'2002 », Ithaca, USA, septembre, 2002, http://www.cs.ucc.ie/~osullb/UICS-02/.

[10] M. DUCASSÉ, L. LANGEVINE. *Analyse automatisée de traces d'exécution de programmes CLP(FD).* in « actes des JFPLC'2002, 11ième Journées francophones de programmation en logique et de programmation par contraintes », Nice, France, juin, 2002.

[11] M. DUCASSÉ, L. LANGEVINE. *Automated Analysis of CLP(FD) Program Execution Traces.* in « Proceedings of the 18th International Conference on Logic Programming (ICLP'02) », série Lecture Notes in Computer Science, volume 2401, Springer-Verlag, éditeurs P. J. STUCKEY., pages 470-471, juillet, 2002.

[12] F. FAGES. *3D Visualization and Control in CLPGUI.* in « Proceedings of UICS'2002, workshop on User Interaction in Constraint Satisfaction, associated to Constraint Programming CP'2002 », Ithaca, USA, Septembre, 2002.

[13] F. FAGES. *CLPGUI : a generic graphical user interface for constraint logic programming over finite domains.* in « Proceedings of the International Workshop on Logic Programming Environments WLPE'02 associated to FLOC'02 », Copenhagen, Danmark, Juillet, 2002.

[14] F. FAGES. *Symbolic model checking of biochemical networks (invited talk).* in « Workshop on Formal Methods and Biological Reasoning, associated to the 3rd International Conference on Systems Biologiy, ICSB'2002 », éditeurs V. SCHÄCHTER., Stockholm, Suède, Décembre, 2002.

[15] G. FERRAND, A. LALLOUET, J. ARSOUZE. *The Constraint System of the Grammatical View of (Constraint) Logic Programming.* in « International Workshop on Functional and (Constraint) Logic Programming », University of Udine, éditeurs M. FALASCHI., Grado, Italy, June, 2002.

[16] G. FERRAND, A. LALLOUET. *A Logic Program characterization of domain reduction approximations in finite domain CSPs.* in « International Conference on Logic Programming », série LNCS, numéro 2401, Springer-Verlag, éditeurs P. J. STUCKEY., pages 478-479, 2002, Poster.

[17] G. FERRAND, W. LESAINT, A. TESSIER. *Theoretical Foundations of Value Withdrawal Explanations for Domain Reduction.* in « Proceedings of the 11th International Workshop on Functional and (Constraint) Logic Programming », série Electronic Notes in Theoretical Computer Science, volume 76, éditeurs M. FALASCHI., Grado, Italy, June, 2002, http://www.elsevier.com/gej-ng/31/29/23/show/Products/notes/index.htt.

[18] L. LANGEVINE, P. DERANSART, M. DUCASSÉ. *Prototypage de traceurs CLP(FD), un modèle de trace et son expérimentation.* in « actes des JFPLC'02, 11ième Journées francophones de programmation en logique et de programmation par contraintes », Nice, France, juin, 2002.

Rapports de recherche et publications internes

[19] N. CHABRIER. *Calcul de processus et processus biologiques.* Rapport de stage de DEA, INRIA, septembre, 2002, http://contraintes.inria.fr/~chabrier/rapport.tgz.

[20] N. CHABRIER, F. FAGES. *Symbolic model checking of biochemical networks.* Rapport de Recherche, INRIA, Novembre, 2002, http://pauillac.inria.fr/~fages/Papers/CF02cmsb.ps.

[21] E. COQUERY, F. FAGES. *TCLP : overloading, subtyping and parametric polymorphism made practical for constraint logic programming.* Rapport de Recherche, INRIA, Mai, 2002,

http://pauillac.inria.fr/~fages/Papers/CF02iclp.ps.

[22] R. HAÆMMERLÉ. *Extension de GNU-Prolog par coroutines.* Projet de fin d'étude d'ingénieur, INRIA et ESIEE Paris, Septembre, 2002.

[23] R. KADDOUCHE. *L'implantation des contraintes globales en GNU-Prolog.* Rapport de stage de DEA, INRIA, Septembre, 2002.

[24] S. KALLA. *Trace générique pour CHR sur domaines finis.* Rapport de DEA, Orléans, INRIA, Septembre, 2002.

[25] A. LAL. *A constraint programming approach to solving the cutset problem.* Rapport de stage, INRIA et IIT, Juillet, 2002.

Divers

[26] A. AGGOUN, P. DERANSART, R. MARTIN. *Débogage dynamique de programmes avec contraintes sur les domaines finis : état et perspectives.* Janvier, 2002, http://contraintes.inria.fr/OADymPPaC, Réalisation D3.3.1.1 (rapport public).

[27] E. COQUERY. *TCLP 0.3 documentation.* Octobre, 2002, http://contraintes.inria.fr/~coquery/tclp.

[28] P. DERANSART, F. FAGES, N. JUSSIEN, L. LANGEVINE, R. MARTIN. *Débogage dynamique de programmes avec contraintes sur les domaines finis : état et perspectives.* Février, 2002, http://contraintes.inria.fr/OADymPPaC, Réalisation D2.1.1.1 (rapport public).

[29] F. FAGES. *CLPGUI 2.1.3 documentation.* Septembre, 2002, http://contraintes.inria.fr/~fages/CLPGUI.

[30] OADYMPPAC. *Premier rapport d'avancement.* Février, 2002, http://contraintes.inria.fr/OADymPPaC, édité par Pierre Deransart.

[31] S. SOLIMAN. *CLP Implementation of a Phase Model Checker.* Short presentation LICS'02 : 17th Annual Symposium on Logic In Computer Science, Short Session, 2002, http://contraintes.inria.fr/~soliman/publi/lics02.ps.gz.