# Avant-Projet OSCAR

## Outils Syntaxiques pour la Construction et l'Analyse de pRogrammes

**Localisation** : *Rocquencourt*

**Mots-clés** : OSCAR, grammaire attribuée, attribut sémantique, analyse de programme, génie logiciel, génération de compilateur, langage de programmation, optimisation de code, programmation fonctionnelle, sémantique, spécification formelle.

## 1 Composition de l'équipe

**Responsable scientifique**

Martin Jourdan, directeur de recherche INRIA

**Responsable permanent**

François Thomasset, directeur de recherche INRIA

**Secrétaire**

Muriel de Bianchi, INRIA, partagée avec AIR, Estime et Ondes

**Personnels INRIA**

Didier Parigot, chargé de recherche

**Chercheur extérieur**

Gilles Roussel, Maître de Conférences, Université de Marne-la-Vallée

**Chercheur post-doctorant**

Uwe Aßmann, Université de Karlsruhe, R.F.A., jusqu'au 31.07.96

**Chercheurs doctorants**

Philippe Canalda, boursier MESR, Université d'Orléans, jusqu'au 15.10.96
Étienne Duris, boursier MESR, Université d'Orléans

**Stagiaires**

Stéphane Leibovitsch, stagiaire de DEA, Université de Paris VII, du 15.03.96 au 30.09.96
Loïc Correnson, stagiaire d'option, École Polytechnique, du 01.04.96 au 05.07.96

# 2 Présentation de l'équipe

L'avant-projet OSCAR est issu de la scission de l'action CHARME selon ses deux axes (cf. le rapport d'activité 1995), l'autre moitié formant l'essentiel de l'avant-projet A3.

Après la fin du projet ESPRIT COMPARE, qui fut une réussite industrielle et "communautaire" mais au cours duquel les travaux de recherche plus fondamentale furent fortement ralentis, 1996 fut pour OSCAR une année de "retour aux sources" et de mutations.

Nous avons décidé d'orienter nos efforts de recherche sur l'étude des divers paradigmes de programmation — grammaires attribuées bien entendu, puisque c'est là que se situe l'essentiel de notre culture, mais aussi programmation fonctionnelle, programmation logique, programmation orientée objet, etc. — , sur le plan des concepts et sur celui des implantations, en vue de faciliter leur comparaison, de favoriser leur interopérabilité et de susciter des synergies entre leurs promoteurs. L'objectif à long terme est de contribuer à la rationalisation du processus de programmation, à travers le langage de programmation. Dans un premier temps, nous nous concentrons sur l'étude fine des grammaires attribuées et de leurs relations avec la programmation fonctionnelle et la programmation orientée objet.

Rappelons que les grammaires attribuées (GA) sont un paradigme de programmation déclaratif et dirigé par la syntaxe, particulièrement adapté au développement de grosses applications à base syntaxique forte, comme les compilateurs et les outils de réingéniérie de logiciel. Depuis longtemps, nous avons produit une contribution très importante à ce domaine, que ce soit sur le plan de la théorie — extensions du concept, en particulier en ce qui concerne la puissance d'expression, la modularité et la généricité —, de l'implantation — analyses statiques performantes, transformation de GA, méthodes d'évaluation concurrentes, etc. — et des applications — développement du système FNC-2, intégration à la plateforme de construction de compilateurs COSY développée dans COMPARE.

# 3 Actions de recherche

## 3.1 Grammaires attribuées : travaux fondamentaux

### 3.1.1 Les GA dynamiques

*Participants* : Didier Parigot, Martin Jourdan, Gilles Roussel, Étienne Duris

En 1995, nous avions introduit les GA dynamiques, une extension du *langage* que constituent les grammaires attribuées qui permet de tirer pleinement parti de la puissance d'expression du *concept* sous-jacent, en utilisant exactement les mêmes techniques d'implantation que pour les GA traditionnelles. Dans une GA dynamique, la grammaire décrit l'ensemble des flots de contrôle possibles, qui peuvent être strictement liés à des arbres physiques, comme dans les GA traditionnelles, ou bien en être arbitrairement découplés. Dans ce contexte, une production décrit un schéma récursif élémentaire (flot de contrôle), tandis que les règles sémantiques décrivent les calculs associés à ce schéma (flot de données). On peut alors décrire des calculs sur des structures qui ne sont pas uniquement des arbres, mais aussi des formes abstraites permettant de décrire des structures infinies, voire des données non structurées.

Notre première présentation des GA dynamiques [210] était, volontairement, assez informelle et intuitive. En 1996, nous avons formalisé la notion de GA dynamique, ainsi que leur implantation à base de séquences de visite [217, 211] ; nous avons en particulier prouvé formellement la validité de la technique de "changement de plan", qui permet de réutiliser sans modification un générateur d'évaluateurs existant, comme celui de notre système FNC-2.

Nous travaillons actuellement sur la définition formelle de la sémantique des GA dynamiques.

### 3.1.2 Généricité dans les GA

*Participants* : Loïc Correnson, Didier Parigot

Nous avons implanté dans le système FNC-2 nos travaux de 1993-94 sur la généricité dans les grammaires attribuées, en réalisant un "instancieur" de GA. Celui-ci prend en entrée un gène, c'est-à-dire une GA décrivant un certain algorithme sur une syntaxe minimale, une syntaxe cible et une description de l'association entre les éléments des deux syntaxes, et produit une nouvelle GA qui décrit l'exécution de l'algorithme du gène sur la syntaxe cible.

Au cours de cette réalisation, L. Correnson a amélioré l'applicabilité de la méthode en introduisant la notion de *forme standard* de la syntaxe d'une GA, ainsi qu'un algorithme de réduction d'une GA en sa forme standard qui préserve la sémantique. Il a aussi prouvé qu'un gène transformé en sa forme standard est instanciable sur un nombre au moins aussi élevé (et en général plus élevé) de syntaxes cible que lorsqu'il est sous sa forme originelle. Il est à noter que cette notion de forme standard peut aussi servir à déterminer (avec des critères syntaxiques) si deux GA sont équivalentes. Par ailleurs, il a donné une nouvelle définition, plus large, de la relation de correspondance entre deux syntaxes.

Ces travaux sont décrits dans [214].

## 3.2 Relations entre les grammaires attribuées et la programmation fonctionnelle

### 3.2.1 Programmation dirigée par la structure et déforestation

*Participants* : Étienne Duris, Didier Parigot, Gilles Roussel, Martin Jourdan

Les GA dynamiques nous permettent d'atteindre le même pouvoir d'expression que les langages fonctionnels du premier ordre. Dans ceux-ci, le problème de l'élimination des structures intermédiaires (déforestation) est abordé à l'aide de l'opérateur de contrôle générique *fold*, qui permet de rendre la structure des données plus explicite (programmation *dirigée par la structure*). Cette propriété structurelle étant intrinsèque aux GA, nous avons été amenés à comparer les GA et les folds, tant sur le plan du pouvoir d'expression que sur celui de leurs méthodes de "composition optimisante" (composition descriptionnelle et fusion, respectivement).

Dans [215, 208, 209], nous présentons nos premiers résultats. Les folds du premier ordre sont équivalents aux GAs purement synthétisées. Dans ce cadre, la fusion des folds et la composition descriptionnelle aboutissent au même résultat, mais cette dernière est complètement indépendante de l'évaluation (transformation de source à source) tandis que la fusion nécessite la connaissance de la méthode d'évaluation. De plus, beaucoup de programmes qui nécessitent l'usage de folds d'ordre supérieur peuvent être traités avec des GA du premier ordre, grâce à la notion d'attribut hérité. Les approches différentes de ce problème dans ces deux formalismes permettent d'envisager des fertilisations croisées entre GA et programmation fonctionnelle.

Plus généralement, les folds étant des *catamorphismes*, nous avons commencé à étudier les GA en termes algébriques, afin de comparer plus facilement les résultats obtenus en GA et en programmation fonctionnelle et de faciliter le passage de connaissances d'un domaine à l'autre.

### 3.2.2 Évaluation indulgente

*Participants* : Étienne Duris, Didier Parigot, Martin Jourdan

Certains exemples de programmes fonctionnels se traitent en "une passe" en évaluation paresseuse mais nécessitent une décomposition en plusieurs fonctions en cas d'évaluation stricte. En général, cette décomposition doit être effectuée à la main. Or ces exemples s'écrivent en GA aussi facilement que la version fonctionnelle paresseuse tandis que la décomposition (calcul de l'ordre d'évaluation et des séquences de visite) est réalisée automatiquement par le système.

Nous avons démarré une étude systématique sur ce sujet, mais nous n'avons pas encore de résultat méritant publication.

### 3.2.3 Sémantique dénotationnelle

*Participants* : Stéphane Leibovitsch, Didier Parigot

Les relations entre la sémantique dénotationnelle et les GA ont été peu étudiées : un seul article de H. Ganziger (1980) traite vraiment de ce sujet, en montrant comment traduire une spécification de langage en sémantique dénotationnelle en un interprète écrit en GA. Malheureusement, les GA produites dépassent le cadre traditionnel des GA, puisqu'elles nécessitent une évaluation paresseuse et circulaire. Or, les GA dynamiques permettent de décrire ce type d'extension. Nous avons donc étudié si elles pouvaient constituer une réponse aux diverses questions laissées ouvertes par Ganzinger.

S. Leibovitsch a su répondre positivement à cette question [216]. Il a repris et adapté le schéma de traduction de Ganzinger et montré que les GA qu'il produit sont des GA dynamiques acceptables par FNC-2. Il a en outre réalisé (au moins en partie) un prototype de traducteur automatique mettant en œuvre ce schéma. La seule construction qui pose encore problème est l'utilisation de continuations pour traiter les transferts de flot de contrôle non structurés.

## 3.3 Grammaires attribuées : le système FNC-2

*Mots-clés* : génération de compilateur.

Depuis plusieurs années maintenant, la plupart de nos travaux sur les grammaires attribuées trouvent leur réalisation dans le système de traitement de GA appelé FNC-2. Rappelons que notre objectif à long terme est d'offrir un système de qualité industrielle, c'est-à-dire puissant, efficace et d'utilisation facile et souple.

Dans les deux applications ci-dessous, nous avons pu, à nouveau, apprécier le haut degré de productivité apporté par les GA et par FNC-2 dans les applications de traitement de langages.

### 3.3.1 Évolution du système FNC-2

*Participant* : Didier Parigot

À part la réalisation de l'instancieur de GA (cf. la section 3.1.2), le système FNC-2 n'a subi que des évolutions mineures. La plus significative est la possibilité de déclarer des GA comme des fonctions et de les utiliser comme telles dans les règles sémantiques. Par ailleurs, quelques bogues ont été corrigés.

### 3.3.2 Un compilateur Java

*Participant* : François Thomasset

*Mots-clés* : Java.

Une maquette de compilateur Java a été réalisée à l'aide de FNC-2. Elle spécifie la construction des tables des symboles, le typage des variables, ainsi que la recherche des méthodes. Une étude de faisabilité d'un compilateur complet a été effectuée.

## 3.4 Optimisation de programmes par réécriture de graphes

*Participant* : Uwe Aßmann

*Mots-clés* : optimisation de code, réécriture de graphe.

U. Aßmann, de l'Université de Karlsruhe (R.F.A.), a achevé cet été son séjour post-doctoral dans notre équipe. Ses travaux concernent une méthode originale de spécification et de mise en œuvre d'analyses et de transformations (optimisations) de programmes, fondée sur les systèmes de réécriture de graphes, soit à ajout d'arcs, soit stratifiés.

Son travail a progressé sur le plan théorique aussi bien que pratique. Sur le premier plan, un nouveau critère de terminaison et la méthode de stratification des systèmes de réécriture de graphes ont été formalisés [212]. Ces méthodes sont à la base de l'applicabilité des systèmes de réécriture de graphes à l'optimisation de programmes. Cet article a été soumis au journal TOPLAS. Enfin, un résumé de la méthode, accompagné des résultats de quelques expériences pratiques, a été publié [207].

Sur le plan pratique, une nouvelle version du générateur d'analyseurs et d'optimiseurs de programmes OPTIMIX, utilisable avec C, a été rendue publique. Quelques fonctionnalités importantes ont été ajoutées au langage de spécification (preuve de terminaison, groupes de règles imbriquées, *view rules*) [213]. Par ailleurs, OPTIMIX est maintenant disponible sur le réseau[1].

## 3.5 Un système de réécriture d'arbres pour la sélection d'instructions

*Participant* : Philippe Canalda, François Thomasset

*Mots-clés* : génération de code, génération de compilateur.

P. Canalda, avec le soutien actif de F. Thomasset, a achevé cette année sa thèse[2] [206] consacrée à la génération automatique de sélecteurs d'instructions à partir de la description de la sémantique des instructions du processeur cible en termes de la représentation intermédiaire. Cette approche est plus agréable pour l'utilisateur (écrivain du compilateur) que celle fondée sur la description directe du processus de génération de code et a été utilisée avec succès dans le système PAGODE, validé dans le projet ESPRIT COMPARE.

La production de code débute par la sélection d'instructions mais comprend aussi l'ensemble des autres tâches constituant la partie arrière d'un compilateur (allocation de registres, réordonnancement, ...). C'est pourquoi le choix de la séquence de code à émettre peut ne pas être optimal dès la phase de sélection d'instructions. Un premier pas vers une génération de code globalement optimale passe par un calcul exhaustif de toutes les façons de traduire une forme intermédiaire donnée vers une forme machine. C'est à la tâche de réordonnancement, qui possède toutes les informations nécessaires, que revient ultimement le choix, parmi toutes ces possibilités, de celle qui est considérée comme la meilleure.

La contribution du sélecteur d'instructions est fondée sur un nouveau système de réécriture d'arbres ascendant (NSRA), inventé pour l'occasion. Un tel formalisme prend en compte la multiplicité des solutions inhérente au problème de sélection d'instructions et rend possible le calcul exhaustif de toutes ces solutions (séquences d'instructions). Les solutions équivalentes à un réordonnancement près sont représentées par une seule structure arborescente. Calculer l'ensemble des solutions revient maintenant à calculer l'ensemble des structures arborescentes correspondant à chaque classe d'équivalence. Ce calcul demeure de complexité exponentielle, mais nous montrons qu'il peut s'effectuer en une seule passe d'analyse ascendante sur le terme candidat à la réécriture. Cette approche est donc adaptée à la génération automatique de sélecteurs d'instructions opérationnels. De plus, elle peut permettre de traiter d'autres problèmes exprimables en termes de traduction d'arbres.

[1] http://i44www.info.uni-karlsruhe.de/ assmann/optimix.html

[2] La soutenance est prévue au printemps 1997.

Dans PAGODE, le sélecteur d'instructions prend en entrée un ensemble de règles de transformation d'arbres calculées à partir d'une spécification de la machine cible écrite en SCALA. Ces règles — qui peuvent être non linéaires — vérifient une propriété PUP (Propriété Une Passe). À cette condition, l'ensemble des règles constitue un NSRA. À partir de celui-ci, nous construisons statiquement un automate d'arbres qui guide le processus dynamique de génération de code en une seule passe. Pour cela, on commence par construire l'automate ascendant de reconnaissance de chaque règle $r_i = (g_i \rightarrow d_i)$ du NSRA, c'est-à-dire un automate de filtrage ascendant du modèle (non linéaire) $g_i$. Ensuite, nous unifions statiquement chaque modèle de sortie $d_i$ avec les états des automates de filtrage des règles du NSRA qui reconnaissent $d_i$. À l'aide d'un tel automate, les applications successives de deux règles de réécriture sur le terme $T$ à réécrire ne requièrent qu'une seule passe ascendante sur la partie de $T$ concernée. Des heuristiques de choix (comme la consommation en cycles d'horloges, la consommation d'espace mémoire, le nombre de ressources utilisées, ...) viennent arbitrer le calcul et la génération d'un ensemble plus réduit de séquences de code.

# 4 Actions industrielles

Nous entretenons des relations suivies avec nos ex-partenaires du projet COMPARE, en particulier ACE (Amsterdam, Pays-Bas) et Steria (Vélizy). P. Canalda travaille chez ACE depuis novembre 1996.

Nous avons entamé en décembre des discussions avec la société Metaware Technologies (Le Pecq). Nous espérons amorcer avec eux une collaboration sur les outils de manipulation de programmes dont ils ont besoin pour leurs applications de réingéniérie de logiciel (migration, adaptation à l'an 2000, etc.).

# 5 Actions nationales et internationales

## 5.1 Bibliographie sur les grammmaires attribuées

Nous tenons à jour et publions sur le Web une bibliographie quasi-exhaustive sur les grammmaires attribuées (environ 900 références) [218]. Les réactions que nous recueillons à son sujet montrent qu'elle est un outil très utilisé et très apprécié. Plus généralement, nous nous efforçons de réunir la communauté internationale des grammmaires attribuées autour de notre site Web[3].

## 5.2 Chercheurs invités

Uwe Aßmann a achevé cet été son séjour post-doctoral dans notre équipe. Ses activités sont décrites dans la section 3.4. Nous restons en contact suivi.

**Séminaires** Uwe Aßmann (INRIA et Université de Karlsruhe, R.F.A.) : *Stratified Graph Rewrite Systems for Program Optimization*, 2 avril 1996 — John Boyland (University of California, Berkeley) : *Conditional Attribute Grammars*, 18 juillet 1996 — Ralf Lämmel (Universität Rostock, R.F.A.) : *Motivating Operations on Attribute Grammars*, 15 novembre 1996.

## 5.3 Contacts universitaires et industriels

Dans le cadre de nos activités de recherche fondamentale, nous entretenons des contacts suivis avec les personnes suivantes : Isabelle Attali (INRIA Sophia-Antipolis), Paul Franchi-Zannettacci (ESSI, Sophia-Antipolis), Françoise Bellegarde (Université de Franche-Comté, Besançon), Doaitse Swierstra et Erik Meijer (Rijksuniversiteit Utrecht, Pays-Bas), John Boyland (University of California, Berkeley,

[3] http://www-rocq.inria.fr/oscar/

USA), Guy Tremblay (Université du Québec à Montréal, Canada), David Barnard (University of Regina, Saskatchewan, Canada), Jon Riecke et John Reppy (AT&T Bell Laboratories, Murray Hill, NJ, USA), Dan Yellin (IBM Research, Hawthorne, NJ, USA), Günter Riedewald et Ralf Lämmel (Universität Rostock, R.F.A.), Stefano Crispi-Reghizzi (Politecnico di Milano, Italie).

Dans le cadre d'activités plus appliquées, en particulier autour du système FNC-2, nous entretenons des contacts suivis avec les personnes et entreprises suivantes : Isabelle Attali (INRIA Sophia-Antipolis), Hubert Garavel (INRIAlpes), Steria (Vélizy). Voir aussi la section 4.

### 5.4 Activités éditoriales

F. Thomasset est membre du comité de rédaction de la revue *Technique et Science Informatiques* (*TSI*) de l'AFCET.

M. Jourdan écrit régulièrement des critiques pour la revue *Computing Reviews* de l'ACM.

## 6 Diffusion des résultats

### 6.1 Actions d'enseignement

#### 6.1.1 Enseignement universitaire

Le projet fait partie de l'Équipe Doctorale du D.E.A. d'Informatique "Langages, programmation, traduction et intelligence artificielle" de l'Université d'Orléans. M. Jourdan et F. Thomasset y enseignent, avec A. Despland, le module "Compilation avancée".

M. Jourdan est Maître de Conférences d'exercice partiel à l'École Polytechnique, où il assure des Petites Classes d'Initiation à l'Informatique ainsi qu'une partie de cours de compilation en majeure "Informatique fondamentale et applications". D. Parigot assure, en tant que Chef de Travaux Pratiques, les Travaux Dirigés correspondants.

É. Duris assure des Travaux Dirigés de langage C en deuxième année de DEUG SM à l'Université de Marne-la-Vallée, ainsi que les Cours/Travaux Dirigés d'initiation aux logiciels de gestion en DESS DPMS-Bio à l'Université d'Orléans.

Ph. Canalda a assuré des Travaux Dirigés en Algorithmique et Structure de Données en première année de l'IUT d'Orléans.

#### 6.1.2 Jurys de thèse

M. Jourdan a été président du jury de la thèse de Sylvain Lelait, intitulée *Contribution à l'allocation des registres dans les boucles*, soutenue le 16 janvier 1996 à l'Université d'Orléans.

### 6.2 Conférences et séminaires

M. Jourdan a participé à la conférence POPL'96 à St-Petersburg Beach (FL, USA) du 21 au 24 janvier. Il a aussi participé aux conférences PLDI'96 et ICFP'96 à Philadelphie (PA, USA) du 22 au 26 mai ; il a ensuite présenté les travaux du projet sur les grammaires attribuées dynamiques [217] aux Bell Laboratories (Murray Hill, NJ, USA) le 28 mai et à IBM Research (Hawthorne, NY, USA) le 29 mai.

D. Parigot a présenté les grammaires attribuées dynamiques à la conférence JFLA'96 (Val-Morin, Québec, Canada, 28-30 janvier) [210] et à la Queen's University (Kingston, Ontario, Canada) le 31 janvier.

D. Parigot, É. Duris et G. Roussel ont présenté les grammaires attribuées dynamiques à la conférence PLILP'96 (Aachen, RFA, 25-27 septembre) [211].

Tous les membres du projet ont participé aux journées du GDR "Programmation" (Orléans, 20-22 novembre). L. Correnson y a présenté ses travaux sur la généricité dans les grammaires attribuées [214] et É. Duris les travaux sur les rapports entre GA et *folds* [208].

U. Aßmann a présenté ses travaux sur l'optimisation de programmes à l'aide de systèmes de réécriture de graphes à la conférence CC'96 (Linköping, Suède, 24-26 avril) [207], à la réunion du groupe 2.4 de l'IFIP qui s'est tenue à Ameland (Pays-Bas) du 3 au 7 juin et à l'Université de Genève (Suisse) le 28 juin.

## 6.3 Organisation de manifestations

M. Jourdan a été membre du comité de programme de CC'96 (*6th Int. Conf. on Compiler Construction*, Linköping, Suède, 22–26 avril 1996).

F. Thomasset a organisé avec C. Eisenbeis (avant-projet A3) les *Rencontres sur les problèmes linéaires en nombres entiers*[4], du 28 au 30 octobre à Rocquencourt.

## 6.4 Diffusion des produits et du savoir-faire

### A l'INRIA, dans l'enseignement et dans la recherche

Le projet Spectre (INRIA Rhône-Alpes)] a effectué une étude d'utilisabilité sur le système FNC-2 et a décidé de le retenir pour servir de base de développement d'un futur compilateur Extended LOTOS.

Par ailleurs, nos produits (essentiellement FNC-2) sont diffusés dans plusieurs instituts de recherche, écoles et universités, à des fins d'enseignement et de recherche. Le système FNC-2 est accessible par FTP[5].

# 7 Publications

## Thèses

[206] P. CANALDA, *Un système de production de sélecteurs d'instructions optimisants fondé sur la réécriture d'arbres pour* PAGODE, thèse de doctorat, Département d'Informatique, Université d'Orléans, mars 1997.

## Communications à des congrès, colloques, etc.

[207] U. ASSMANN, «How To Uniformly Specify Program Analysis and Transformation», *in : 6th Int. Conf. on Compiler Construction (CC'96)*, P. A. Fritzson (réd.), *Lect. Notes in Comp. Sci.*, *1060*, Springer-Verlag, avril 1996.

[208] E. DURIS, D. PARIGOT, G. ROUSSEL, M. JOURDAN, «Grammaires Attribuées et Folds : Opérateurs de Contrôle Génériques», *in : Journées du GDR Programmation*, Orléans, novembre 1996. Fichier PostScript comprimé[6].

[209] E. DURIS, D. PARIGOT, G. ROUSSEL, M. JOURDAN, «Grammaires Attribuées et Folds : Opérateurs de Contrôle Génériques», *in : Journées Francophones des Langages Applicatifs (JFLA'97)*, *Collection didactique*, INRIA, Dolomieu, janvier 1997. Fichier PostScript comprimé[7].

[4] http://www-rocq.inria.fr/ thomasse/plne.html

[5] ftp://ftp.inria.fr/INRIA/Projects/oscar/FNC-2/

[6] ftp://ftp.inria.fr/INRIA/Projects/oscar/FNC-2/publications/gdr96.ps.gz

[7] ftp://ftp.inria.fr/INRIA/Projects/oscar/FNC-2/publications/jfla97.ps.gz

[210] D. PARIGOT, É. DURIS, G. ROUSSEL, M. JOURDAN, « Les grammaires attribuées : un langage fonctionnel déclaratif », *in : Journées Francophones des Langages Applicatifs (JFLA'96)*, G. Lapalme, C. Queinnec (réd.), *Collection didactique*, *15*, INRIA, p. 263–279, Val-Morin, Québec, janvier 1996. Fichier PostScript comprimé[8].

[211] D. PARIGOT, G. ROUSSEL, M. JOURDAN, É. DURIS, « Dynamic Attribute Grammars », *in : Int. Symp. on Programming Languages, Implementations, Logics and Programs (PLILP'96)*, H. Kuchen, S. D. Swierstra (réd.), *Lect. Notes in Comp. Sci.*, *1140*, Springer-Verlag, p. 122–136, Aachen, septembre 1996. Fichier PostScript comprimé[9].

### Rapports de recherche et publications internes

[212] U. ASSMANN, « Graph Rewrite Systems For Program Optimization », *rapport de recherche n°2955*, INRIA, juillet 1996, soumis à TOPLAS. Fichier PostScript comprimé[10].

[213] U. ASSMANN, « OPTIMIX Language Manual (for OPTIMIX 2.0) », *rapport technique n°195*, INRIA, juillet 1996, Fichier PostScript comprimé[11].

[214] L. CORRENSON, « Généricité dans les Grammaires Attribuées », *Rapport de stage d'option*, École Polytechnique, juillet 1996, Fichier PostScript comprimé[12].

[215] E. DURIS, D. PARIGOT, G. ROUSSEL, M. JOURDAN, « Attribute Grammars and Folds: Generic Control Operators », *rapport de recherche n°2957*, INRIA, août 1996, Fichier PostScript comprimé[13].

[216] S. LEIBOVITSCH, « Relations entre la sémantique dénotationnelle et les grammaires attribuées », *Rapport de DEA*, Université de Paris VII, septembre 1996, Fichier PostScript comprimé[14].

[217] D. PARIGOT, G. ROUSSEL, M. JOURDAN, É. DURIS, « Dynamic Attribute Grammars », *rapport de recherche n°2881*, INRIA, mai 1996, Fichier PostScript comprimé[15].

### Divers

[218] D. PARIGOT, M. JOURDAN, « A Bibliography on Attribute Grammars », Page HTML avec outil de recherche[16], mise à jour régulièrement. Contient environ 900 références à des publications sur les grammaires attribuées.

## 8 Abstract

The OSCAR group studies various programming paradigms in order to expose and exploit their possible synergies at both the conceptual and implementation levels. The first and main subject of these studies is attribute grammars (AGs) and their relationship with functional and object-oriented programming. In 1996 the main results were: formal definition and implementation of dynamic AGs; implementation

[8] ftp://ftp.inria.fr/INRIA/Projects/oscar/FNC-2/publications/jfla96.ps.gz

[9] ftp://ftp.inria.fr/INRIA/Projects/oscar/FNC-2/publications/plilp96.ps.gz

[10] ftp://ftp.inria.fr/INRIA/publication/RR/RR-2955.ps.gz

[11] ftp://ftp.inria.fr/INRIA/publication/RT/RT-0195.ps.gz

[12] ftp://ftp.inria.fr/INRIA/Projects/oscar/FNC-2/publications/Rapport/correnso.ps.gz

[13] ftp://ftp.inria.fr/INRIA/publication/RR/RR-2957.ps.gz

[14] ftp://ftp.inria.fr/INRIA/Projects/oscar/FNC-2/publications/Rapport/leibovit.ps.gz

[15] ftp://ftp.inria.fr/INRIA/publication/RR/RR-2881.ps.gz

[16] http://www-rocq.inria.fr/oscar/FNC-2/AG.html

and improvement of the instanciation of generic AGs; comparison between AGs and the *fold* operator of functional programming, regarding both their expressive power and their function-composition optimization technique; translation of denotational semantics specifications into dynamic AGs; evolution of the FNC-2 AG-processing system; and the development of a Java compiler using FNC-2. Other topics of study include program analysis and transformation by means of graph rewriting, and generation of instruction selectors based on tree rewriting.