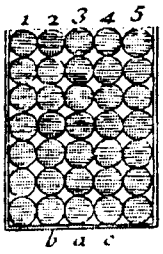
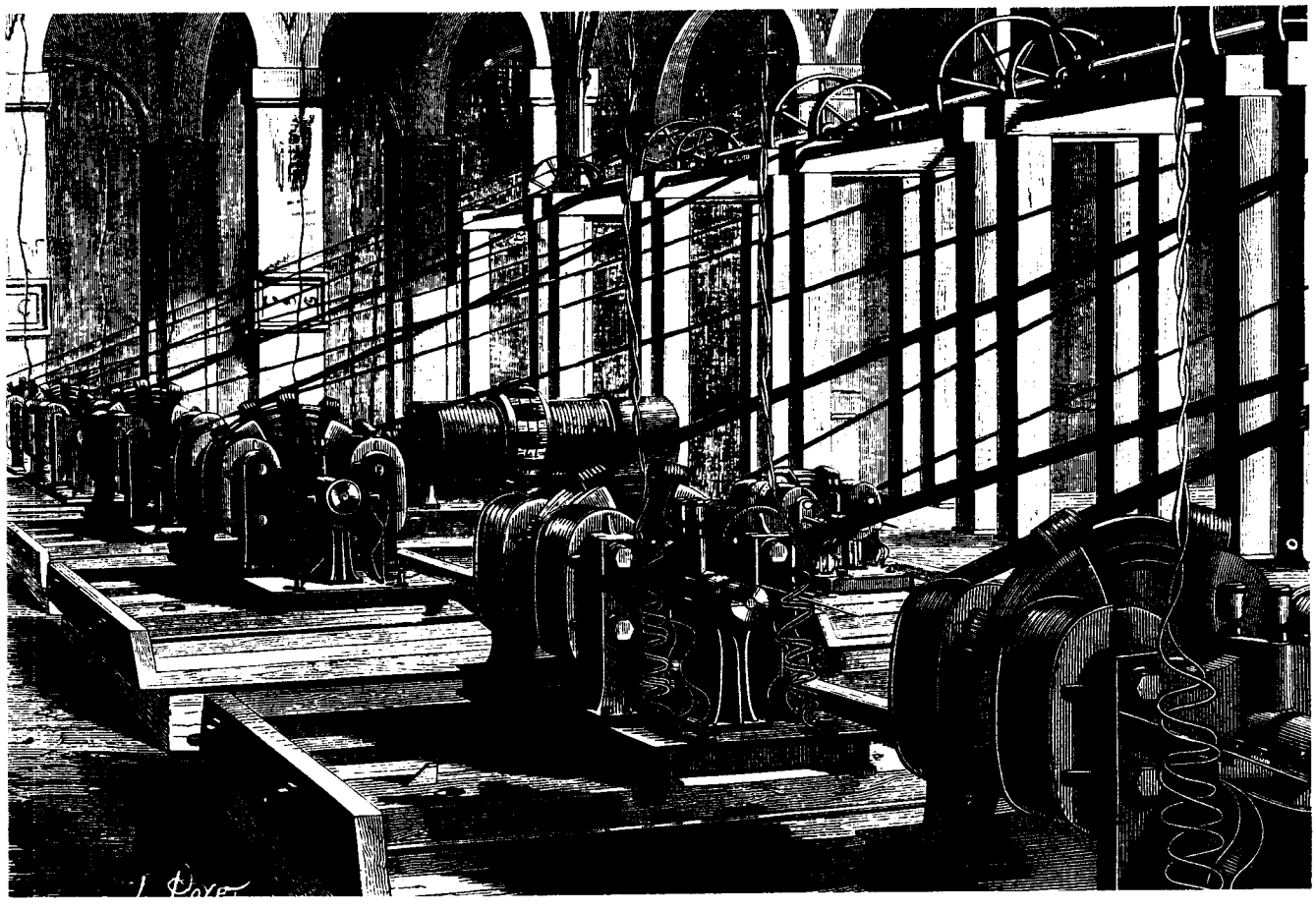


BIGRE

ISSN 0221-5225
BIGRE N° 66
Décembre 1989

Journées AFCET GROPLAN — Chamonix, Mai 1989



PROGRAMMES POUR MACHINES PARALLELES

IRISA
Campus de Beaulieu
F-35042 Rennes cedex

BIGRE

ISSN 0221-5225
BIGRE N° 66
Décembre 1989

Actes des Journées AFCET-GROPLAN

PROGRAMMES POUR MACHINES PARALLELES

Chamonix, Mai 1989

BIGRE N° 66

Rédaction, administration : IRISA, Campus de Beaulieu, 35042 Rennes cedex
Directeur de la publication : Jean-Pierre Verjus
Rédaction : J. André, J. Bézivin, J.L. Nebut et R. Rannou
N° Commission paritaire de presse : 1412 ADEP
Abonnement pour 1990 : 200 francs
Dépot légal : 4ème trimestre 1990
Tirage : 500 exemplaires

GAMMA
Un formalisme pour la construction de programmes parallèles

Jean-Pierre BANATRE Daniel LE METAYER
Louis MUSSAT
Equipe Langages et Systèmes Parallèles
IRISA Rennes

1 Introduction

La définition de méthodes de programmation, domaine de recherche très actif, est une nécéssité cruciale pour la programmation de machines parallèles. Malheureusement, les approches traditionnelles sont mal adaptées à l'expression du parallélisme de haut niveau. Dans le cadre impératif la gestion des flots de contrôle est à la charge du programmeur. Dans le cadre fonctionnel la récursivité des structures de données entraîne celle des programmes, ce qui limite les possibilités de concurrence.

C'est pourquoi nous proposons un nouveau modèle [BLM86, BCLM88], appelé *Gamma*, basé sur la métaphore de la réaction chimique : le calcul peut être vu comme une succession de réactions chimiques consommant et produisant des éléments dans un multi-ensemble. Nous montrons rapidement comment dériver des programmes dans ce formalisme [Cou86, Mus88], insistant plus sur la technique originale de preuve de terminaison.

2 Le modèle Gamma

Le multi-ensemble est une structure de données peu contraignante qui ne diffère de l'ensemble que par la possibilité de contenir plusieurs occurences d'une même valeur. L'opérateur Γ va permettre de transformer un multi-ensemble par application de règles de sélection et de production d'éléments. Formellement :

$$\Gamma(R,A)(M) = \begin{array}{l}\textbf{si } \exists \vec{X}, R(\vec{X}) \\ \textbf{alors } \Gamma(R,A)((M-X)+A(\vec{X})) \\ \textbf{sinon } M\end{array}$$

Le prédicat R est la *condition de réaction*, et indique quand des éléments peuvent réagir. La fonction A, appelée *action*, donne le résultat de la réaction. Les éléments captés par R sont remplacés par les éléments produits par A, le calcul ne s'arrêtant qu'en l'absence de valeurs pouvant réagir. Le contrôle non-impératif autorise une implémentation concurrente; les transformations s'effectuent sur plusieurs n-uplets $\vec{X}$ *indépendants* vérifiant R et choisis arbitrairement. Tous les remplacements peuvent se faire de façon asynchrone, puisqu'aucune propriété globale sur la structure de données n'est à maintenir.

Cette définition n'est pas la plus générale; en fait l'opérateur Γ peut accepter un nombre quelconque de couples (R,A). Chaque condition de réaction indique quand l'action associée peut s'appliquer. Néanmoins les programmes simples peuvent sécrire avec un seul couple (R,A), aussi nous limiterons nous à ce cas.

Il est temps de donner deux exemples.

3 Exemples

3.1 Crible d'Erathostène

La fonction premiers(n) calculant l'ensemble des nombres premiers inferieurs à n s'écrit très simplement en Gamma :

$$\textbf{premiers}(n) = \Gamma(R,A)(\{2,\ldots,n\})$$

$$\textbf{avec} \begin{cases} R(x,y) = y \mid x \\ A(x,y) = \{y\} \end{cases}$$

où $y \mid x$ signifie " y divise x ". La figure 1 montre une évaluation possible de premiers(8).

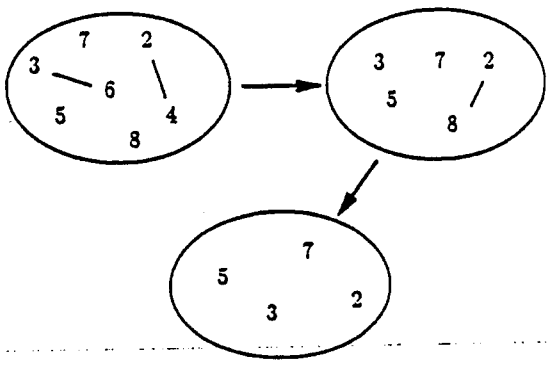


Figure 1 : premiers(8)

3.2 Tri par échanges

Les éléments manipulés sont des couples *(valeur, index)*. L'index donnera à la fin du calcul le rang de la valeur dans l'ordre croissant.

$$\mathbf{tri}(M) = \Gamma(R, A)(M)$$

$$\text{avec} \begin{cases} R(x,y) & = & (x.i < y.i) \wedge (x.v > y.v) \\ A(x,y) & = & \{(x.v, y.i), (y.v, x.i)\} \end{cases}$$

La figure 2 montre une évaluation de $\mathbf{tri}(\{(8,4),(5,2),(9,3),(13,1),(9,5)\})$.

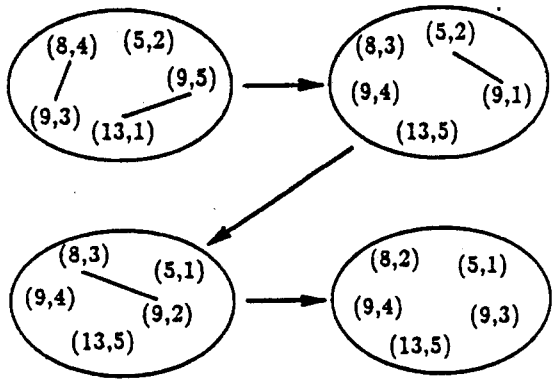


Figure 2 : $\mathbf{tri}(\{(8,4),\ldots,(9,5)\})$

Nous allons maintenant exposer la méthode de dérivation.

4 Dérivation

Le problème à résoudre est défini par une *spécification* écrite dans un langage de logique du premier ordre. Cette spécification décrit les relations du résultat avec les données initiales et ses propriétés. Après avoir mis les formules sous forme normale disjonctive, on les répartit en un *variant* et un *invariant*. Ce dernier, vérifié dans l'état initial, devra être maintenu par le programme dérivé tout au long de son exécution. Ainsi la spécification est satisfaite dès que le variant est établi ; le calcul peut alors s'arrêter. Par négation des formules constituant le variant, on obtient la condition de réaction, puis on définit l'action associée à l'aide de l'invariant. On aura ainsi dérivé un algorithme partiellement correct. Avant d'en venir à la correction totale, nous allons illustrer les premières étapes de la méthode (résumée figure 3) sur un exemple.

4.1 Dérivation du tri par échanges

Nous obtenons le multi-ensemble trié M à partir de M_0 par la spécification suivante :

$$\begin{cases} & M_0 : \{(v\text{: valeur}, i\text{: index})\} \\ & M_0.i = \{1 \ldots \#M_0\} \\ & M : \{(v\text{: valeur}, i\text{: index})\} \\ (1) & M.v = M_0.v \\ (2) & M.i = M_0.i \\ (3) & \forall \{x,y\} \in M,\ x.i < y.i \Rightarrow x.v \leq y.v \end{cases}$$

L'implication

$$M = M_0 \Rightarrow (1) \wedge (2)$$

donne pour invariant et variant

$$I \equiv (1) \wedge (2) \quad \text{et} \quad V \equiv (3)\ .$$

La négation de V,

$$\neg V \equiv \exists \{x,y\} \subseteq M,\ x.i < y.i \wedge x.v > y.v\ ,$$

livre la condition de réaction

$$R(x,y) \equiv (x.i < y.i) \wedge (x.v > y.v)\ .$$

Le maintien de l'invariant impose les égalités

$$\begin{array}{rcl} A(x,y).v & = & \{x.v, y.v\} \\ A(x,y).i & = & \{x.i, y.i\} \end{array}$$

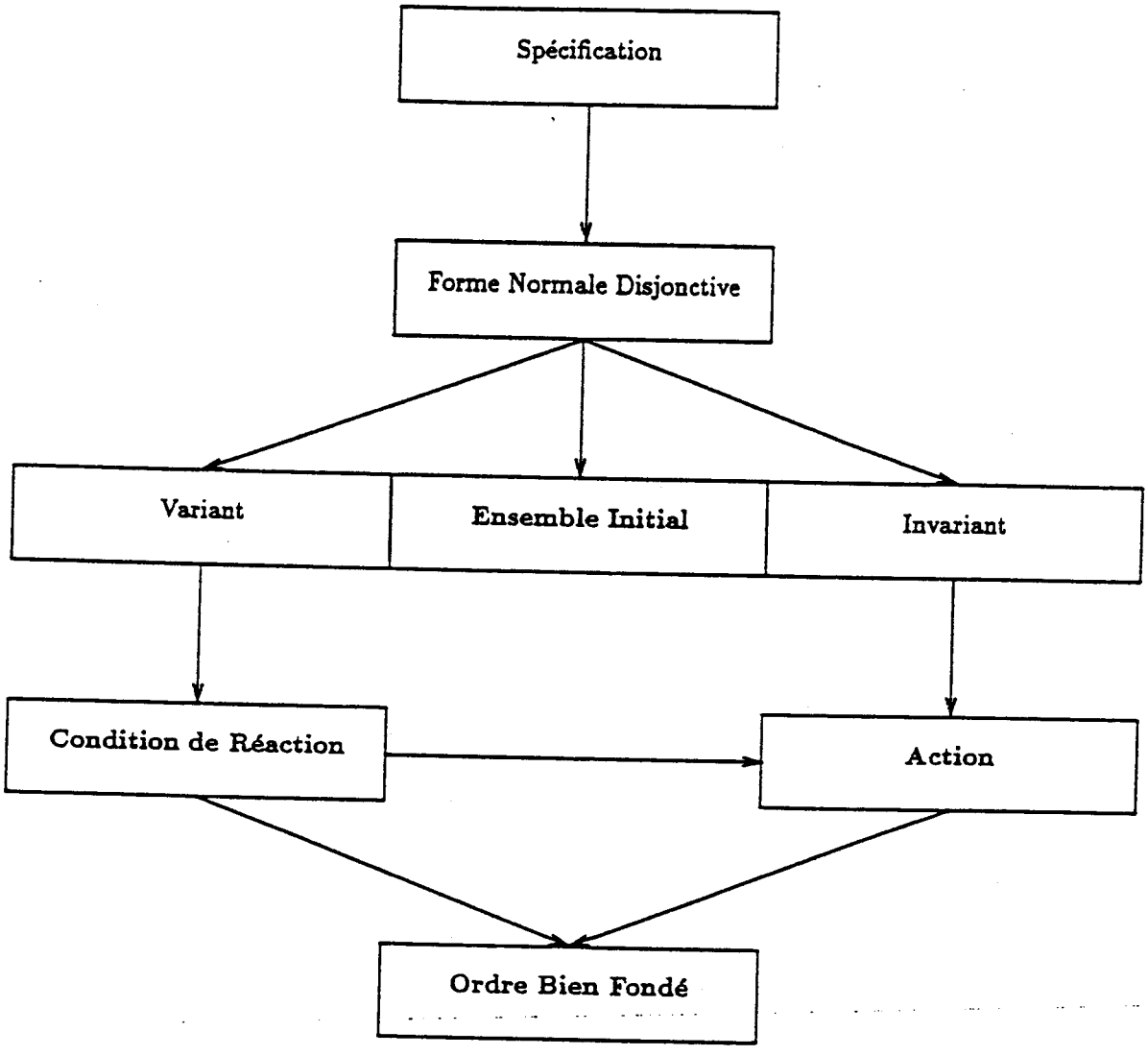


Figure 3 : Shéma de dérivation

et la nécéssité de progrès ($\{x,y\} \neq A(x,y)$) impliquent

$$A(x,y) = \{(x.v, y.i), (y.v, x.i)\}$$

On a obtenu le programme

$$\mathbf{tri}(M) = \Gamma(R,A)(M)$$

$$\text{avec} \begin{cases} R(x,y) &= (x.i < y.i) \wedge (x.v > y.v) \\ A(x,y) &= \{(x.v, y.i), (y.v, x.i)\}. \end{cases}$$

Il reste à démontrer la terminaison de l'algorithme dérivé. On le fait en utilisant les *ordres de Dershowitz - Manna* [DM79]. Ce sont des ordres construits sur un multi-ensemble à partir d'ordres sur ses propres éléments. En voici la définition :

Définition 1 (Ordre $\prec\!\!\prec$)
Soit $(E, \prec)$ un ensemble ordonné. L'ensemble $\mathcal{M}(E)$ des multi-ensembles finis sur E est ordonné par

$$F \prec\!\!\prec G$$

s'il existe deux multi-ensembles X et Y, avec $\emptyset \neq X \subseteq G$, tels que

$$F = G - X + Y$$
$$et \quad \forall y \in Y,\ \exists x \in X,\ y \prec x.$$

On a alors le

Théorème 1

$$(\mathcal{M}(E), \prec\!\!\prec) \textit{ est bien fondé}$$
$$\Leftrightarrow \quad (E, \prec) \textit{ est bien fondé.}$$

Si l'action est telle que $A(\vec{X}) \prec\!\!\prec \vec{X}$, alors $(M - X) + A(\vec{X}) \prec\!\!\prec M$, et on voit que $(\mathcal{M}(E), \prec\!\!\prec)$ bien fondé assure la terminaison du programme. Le problème est ramené à la construction d'un ordre $\prec$ sur les éléments du multi-ensemble manipulé. Voyons comment procéder.

4.2 Construction d'un ordre $\prec$

La condition de réaction R et l'action A étant de la forme :

$$\begin{array}{lll} R: & E^n & \to \mathbb{B} \\ A: & E^n & \to \mathcal{M}(E) \\ & \vec{X} & \mapsto \{f_1(\vec{X}), \ldots, f_m(\vec{X})\}. \end{array}$$

on donne les définitions suivantes :

Définition 2 (Les relations $\sqsubset_\sigma$)
Soit σ une fonction des m premiers entiers dans les n premiers entiers :

$$\sigma : < 1 \cdots m > \longrightarrow < 1 \cdots n > .$$

La relation $\sqsubset_\sigma$ est définie sur E par :

$$a \sqsubset_\sigma b$$
$$\Leftrightarrow \quad \exists \vec{X} \in E^n,\ R(\vec{X}) \wedge \forall i,\ a \neq x_i$$
$$\wedge \exists j,\ a = f_j(\vec{X}) \wedge b = x_{\sigma j}.$$

Chaque élément produit est mis en relation par $\sqsubset_\sigma$ avec un élément saisi, d'une manière qui ne dépend que de la fonction σ choisie.

Définition 3 (Les relations $\prec_\sigma$)
La relation $\prec_\sigma$ est la fermeture transitive de la relation $\sqsubset_\sigma$.

Toute relation $\prec_\sigma$ irreflexive sera alors un ordre strict sur les éléments de E, et on aura :

$$X \subseteq M \wedge R(\vec{X})$$
$$\Rightarrow \quad M - X + A(\vec{X}) \prec\!\!\prec_\sigma M$$

Revenons à l'exemple développé ci-dessus.

4.3 Terminaison du tri par échanges

Choisissant pour σ la fonction $\bar{1}$ constamment égale à 1, nous avons

$$a \sqsubset_{\mathcal{I}} b$$
$$\Leftrightarrow \quad \exists (x,y),\ (x.i < y.i) \wedge (x.v > y.v)$$
$$\wedge a \neq x \wedge a \neq y \wedge$$
$$([a = (x.v, y.i) \wedge b = x] \vee$$
$$[a = (y.v, x.i) \wedge b = x])$$
$$\Leftrightarrow \quad [(\bot_{\mathcal{V}} < a.v = b.v) \wedge (a.i > b.i)]$$
$$\vee [(a.v < b.v) \wedge (\top_{\mathcal{I}} > a.i = b.i)].$$

où $\bot_{\mathcal{V}}$ est l'élément minimal de l'espace des valeurs, et $\top_{\mathcal{I}}$ l'élément maximal de l'espace des index. La fermeture transitive de cette relation est :

$$a \prec_{\mathcal{I}} b$$
$$\Leftrightarrow \quad [(\bot_{\mathcal{V}} < a.v \leq b.v) \wedge (a.i > b.i)]$$
$$\vee [(a.v < b.v) \wedge (\top_{\mathcal{I}} > a.i \geq b.i)]$$

et est réflexive. De plus c'est un bon ordre car $\mathcal{V} \times \mathcal{I}$ est fini.

5 Conclusion

Malgré le caractère succinct de cet exposé, nous espérons avoir montré l'intérêt du modèle de calcul Gamma. Ce modèle a permis l'élaboration d'une méthode de construction de programmes originale, en particulier pour les preuves de terminaison. Les travaux actuels portent sur :

- la formalisation du processus de dérivation ([Mor87]),
- la réalisation d'un système d'aide automatique à la dérivation,
- la réduction de la combinatoire importante introduite par le modèle en vue d'une mise en œuvre efficace sur machine parallèle.

Bibliographie

[BCLM88] J.P. Banâtre, Anne Coutant, and Daniel Le Métayer. "Parallel Machines for Multiset Transformation and their Programming Style". *Informationstechnik*, 30(2):99–109, 1988.

[BLM86] J.P. Banâtre and Daniel Le Métayer. "A new computational model and its discipline of programming". September 1986. *INRIA research report 566.*

[Cou86] Anne Coutant. "Synthèse de Programmes dans le Formalisme Γ". 1986. *Rapport de D.E.A.*

[DM79] Nachum Dershowitz and Zohar Manna. "Proving Termination with Multiset Orderings". *Communications of the ACM*, 22(8):465–476, August 1979.

[Mor87] Joseph M. Morris. "A Theoretical Basis for Stepwise Refinement and the Programming Calculus". *Science of Computer Programming*, 9:287–306, 1987.

[Mus88] Louis Mussat. "Vers un Système de Développement de Programmes Γ". Septembre 1988. *Rapport de D.E.A.*