Programming by Multiset Transformation.

The GAMMA approach

Jean-Pierre Banâtre
Irisa
Campus de Beaulieu
35042 Rennes Cedex - France
Contents

- Some principles
- A bit of history
- Basic ideas
- The Gamma operator
- Programming with Gamma
- Generalized multisets
- Gamma discipline of programming
- Conclusion: towards a Higher-Order Chemical Language
Some Principles

- Logical parallelism vs physical parallelism

- Strong correlation between control structures and data structures
Objective

Show how this simple ideas have been applied in the design of parallel program structures allowing clean and elegant expression of logical parallelism...
A bit of history.

- Parallel programming.
- Events and compilation.
- Multisets of events and compilation.
- Communication by neighborhood.
Basic ideas.

- One data structure: the multiset
- One paradigm: the Chemical Metaphor
- The Gamma operator
What is a multiset? Why?

- Same as a set, but with the possibility of replication.
- As few constraints as possible.
Definition of the $\Gamma$ operator

$$\Gamma \left( ((R_1, A_1), \ldots, (R_m, A_m)) \right) (M) =$$

if $\forall i \in [1,m] \forall x_1, \ldots, x_n \in M, \neg R_i (x_1, \ldots, x_n)$
then $M$
else let $x_1, \ldots, x_n \in M$, let $i \in [1,m]$ such that $R_i (x_1, \ldots, x_n)$ in
$$\Gamma \left( ((R_1, A_1), \ldots, (R_m, A_m)) \right) \left( (M - \{x_1, \ldots, x_n\}) + A_i (x_1, \ldots, n_x) \right)$$
Our syntax.

\[
\text{replace } x_1, \ldots, x_n \quad \text{by } A(x_1, \ldots, x_n) \quad \text{if } C(x_1, \ldots, x_n)
\]

\[
\text{max } = \text{replace } x, y \quad \text{by } x \quad \text{if } x > y
\]
Examples of Gamma programs

Prime number generation

Goal: produce the prime numbers less than a given $N$

Solution:

$$\text{primes}(N) = \text{rem}\{2\ldots N\}$$

$\text{rem}: \text{replace } x, y \text{ by } y \text{ if multiple } (x,y)$
Number sorting

Goal: sort a set of numbers, each number being represented by a pair (index, value)

Solution:

sort: replace \((i, x)\), \((j, y)\) by \((i, y)\), \((j, x)\)

if \((i > j) \land (x < y)\)
Factorial

Goal: compute $N!$

Solution:

$$\text{factorial } (N) = \text{fact } \{2 \ldots N\}$$

$$\text{fact : place } x,y \text{ by } x \ast y \text{ if true}$$
The majority element problem

Goal: compute the majority element of a multiset $M$. This element appears more than $\text{card}(M)/2$ times in the multiset.

Solution:

\[ \text{MAJ} = \text{one of } \text{maj}(M) \]

\[ \text{maj} : \text{replace } x, y \text{ by } \{\} \text{ if } x \neq y \]
Convex hull

Goal: compute the smallest convex polygon containing a set of points in the plane

Solution:

convex: replace $P_1, P_2, P_3, P_4$

by $P_1, P_2, P_3$

if $P_4$ inside $<P_1, P_2, P_3>$
The dining philosophers

Goal : solve the traditional dining philosopher problem

Solution :

\[
\text{phil} : \text{replace } \psi_i, \psi_j \text{ by } \phi_i \text{ if } j = i \oplus 1 \\
\text{replace } \phi_i \text{ by } \psi_i, \psi_i \oplus 1 \text{ if true}
\]
Generalized multisets.

- Infinite multiplicities.
- Negative multiplicities.
- Variable multiplicities
About multiplicities

- **Multiplets**: \( a^5 \equiv a, a, a, a, a \)

- **Infinite multiplicity**
  - Coding by generators (**replace** \( \emptyset \) **by** \( a \)) and lazy evaluation.
  - Semantics of infinity \( a^\infty, a^\infty = a^\infty \) : how to merge generators ?
  - To handle atomically an infinity of copies : **replace** \( x^\infty \) **by** \( x \).
Example: Sorting by pivot.

\[ 5: \text{Pivot}, 3, 8, 6, 9, 2, 5, \]
\[ (\text{replace } p : \text{Pivot, } x \textbf{ by } p : \text{Pivot } \textbf{if } x > p ) \]

\[ 5: \text{Pivot}^\infty, 3, 8, 6, 9, 2, 5, \]
\[ (\text{replace } p : \text{Pivot, } x \textbf{ by } \emptyset \textbf{ if } x > p) \]

\[ \text{replace } x : \text{Pivot}^\infty \textbf{ by } \emptyset \]
Negative multiplicity.

- Negative multiplicity: $a^{-5}$

How to interpret it?

Missing and expected elements (resources).

Destructors
Negative infinite multiplicity.

- Negative infinite multiplicity: the black hole.

\[ X^{-\infty} \]

*replace* \( X^{-\infty} \) *by* \( \emptyset \) (anticipation)

Jean-Pierre Banâtre, Pascal Fradet and Yann Radenac. Chemical Programming with Infinite and Hybrid Multisets (under preparation).
Variable multisets.

- $\text{To}\_\text{set} = \text{replace } x^n \text{ by } x \text{ if } n > 1$

- $\text{Div}\_\text{by}\_y = \text{replace } 1^z \text{ by } "1" \text{ if } z = y$
Gamma discipline of programming

Given a specification (S), the derivation performs as follows (cf Dijkstra’s program calculus):

- split of the specification: $S = I \land T$
  - $I$: invariant
  - $T$: termination condition
Gamma discipline of programming

- Reaction condition: $\neg T$

- Action: maintains I while progressing towards termination (well-founded ordering).
Example: a sorting algorithm

number = (index, value)

Specification:

(1) \( \forall x, y \in M, \ x.i < y.i \Rightarrow x.v \leq y.v \)
(2) \( M.i = \{1\ldots \text{card} (M_0)\} \)
(3) \( M.v = M_0.v \)
Split of the specification

\[ I = (2) \land (3) \]
\[ \text{easily established from } M_0 \]

\[ T = (1) \]
\[ \text{should hold upon termination} \]
Reaction condition

Negation of $T$

$$R(x,y) = (x.i < y.i) \land (x.v > y.v)$$
Action

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

The elements (x.i, y.v) and (y.i, x.v) which are ill-ordered are replaced by {(x.i, y.v) and (y.i, x.v) which are correctly ordered.}
Well-founded ordering

Derivation of a well-founded ordering on multisets from a well-founded ordering on elements.

Let $>$ be an ordering on $S$.

The ordering $>>$ on multisets $M(S)$ is defined as:

$M >> M'$ $\iff \exists X \in M(S), \exists Y \in M(S)$ such that

$(X \neq \{\} \land X \subseteq M$

$\land M' = (M - X) + Y$

$\land \forall y \in Y, \exists x \in X, x > y$

$\hspace{2.5cm}$

$(i, x) \subseteq (i', x')$ $\iff i \geq i' \land x' \geq x$
Conclusion (1)

- the multiset is a very nice (ideal) data structure
- the chemical reaction is a very nice (idea) control structure
Conclusion (2)

- Another approach to programming: global programming.
- Autonomic properties for free:

\[
\text{Stabilisation} \rightarrow \text{perturbation} \rightarrow \text{re-stabilisation}
\]

\[
\langle \text{sort, 1:3, 2:1} \rangle \rightarrow \gamma \langle \text{sort, 1:1, 2:3} \rangle
\]

\[
\langle \text{sort, 1:1, 2:3, 1:2} \rangle \rightarrow \gamma \langle \text{sort, 1:1, 2:2, 3:3} \rangle
\]
Conclusion (3)

- “Gamma: fifteen years after” describes most of the work done on Gamma up to 2003.

- New ideas:
  - Chemical data structures.
  - Gamma-programs in the test tube… Higher-order Gamma!!!