DEA informatique
université Montpellier II

Autour des travaux de J.L. Giavitto

ou

à la recherche de nouveaux concepts pour programmer des machines massivement parallèles

Phil. Reitz - LIRMM (reitz@lirmm.fr)

16 décembre 2002

Plan

Introduction

La démarche

- 1988 : développer un langage de programmation parallèle

  exploiter de bonnes propriétés pour faciliter sa compilation sur des machines massivement parallèles

- 1991 : constater que le principal demandeur est le monde de la simulation, et qu'il a beaucoup à dire sur les structures fondamentales

  principaux demandeurs et bailleurs de fonds : biologie, agronomie, physique fondamentale, …

- 1997 : concilier les deux mondes : informatique / simulation

  slogan : une approche formelle capable d'appréhender les deux mondes est la topologie algébrique (combinatoire)

Simulation de systèmes dynamiques

Problèmes de représentation

- du temps
- de l’espace
- de l’énergie (états)
- des processus, de leurs interactions

Problèmes des échelles multiples

Problèmes de la multiplicité des points de vue

- fonctionnel
- structurel
- causalité
- intégration discret / continu

Guide pour la conception d’un langage dédié

Le langage doit pouvoir servir aussi bien à

- décrire un modèle
- l’opérationnaliser
- le valider

Le langage 8 ½

Les entités manipulées : trois concepts clés

- Flot (*stream*) = structuration du temps
- Collection = structuration de l'espace
- Tissu (*web*) = composition de flots et de collections

  ⇒ processus = programme 8 ½

Une approche déclarative

- Programme = système d'équations sur un tissu

  ⇒ cadre fonctionnel pur

  ⇒ langage typé statiquement, types inférés
- Compilateur : c'est à lui que revient le placement et le séquencement des processus

Des contraintes fortes

- Les collections et les flots sont homogènes
- Les processus évoluent de façon synchrone (horloge globale)

Les flots (*stream*)

Définition

Flot = variable qui change de valeur dans le temps

$\Rightarrow$ suite de valeurs

- Les flots partagent une même référence temporelle (tics)
- Une valeur ne dépend que d'un nombre fini de valeurs passées, à horizon temporel borné

Autrement dit, un flot F est caractérisé par 2 fonctions :

$$\begin{aligned} F &= < \; 0; \quad 1; \quad 2 \quad < \\ \text{horloge}(F) = H_F &= < \; v; \; f; \; v; \; f; \; v; \; f < \\ \text{valeur}(F) = V_F &= < \; 0; \; 0; \; 1; \; 1; \; 2; \; 2 < \end{aligned}$$

top = tic pour lequel l'horloge est vraie (v)

Flots constants

Tout scalaire constant est un flot

$$\begin{aligned} 12 &= < \; 12 \quad < \\ \text{horloge}(12) = H_{12} &= < \; v; \; f; \; f; \; \ldots < \\ \text{valeur}(12) = V_{12} &= < \; 12; \; 12; \; 12; \; \ldots < \end{aligned}$$

Un flot jouant un rôle important (synchronisation) :

$$\begin{aligned} \text{horloge}(\texttt{Clock}) &= < \; v; \; v; \; v; \; \ldots < \\ \text{valeur}(\texttt{Clock}) &= < \; v; \; v; \; v; \; \ldots < \end{aligned}$$

Opérations principales

- Illustrées sur des exemples

Tic Þ	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
X		2	3		2		6		7		8
Y	8		7		5	6	1	1	4		
A	v		f	f	v	v	f		v		f

Extension des opérations classiques

Tic Þ	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
X+Y	⊥	10	10		7	8	7	7	11		12

Retard

Tic Þ	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
$X			2		3		2		6		7

Echantillonnage

Tic Þ	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
X when A					2	2			7		

Exemples de programmes sur les flots

Equations quantifiées

`F@t` : valeur du flot `F` au top n°`t`

```
A   = 5
B   = 6 when Clock
C@0 = 1
C   = A + $B
```

Des exemples classiques

Fibonacci

```
fib@0 = 1
fib@1 = 1
fib   = $fib+$$fib when Clock
```

Factorielle

```
n@0 = 0
n   = 1+$n when Clock

fact@0 = 1
fact   = n*$fact
```

Les collections

Définition

Collection = agrégat de composants

⇒ composants tous d'un même type

⇒ chaque composant est repéré par un index (coordonnée dans l'espace qu'est la collection)

⇒ un composant est soit simple (scalaire, flot), soit une collection

Notations

Construction

`{ 3, 7, 5, 9 }` de type `int[4]`

Projection

`{ 3, 7, 5, 9 }.2` vaut `5`

Principales opérations

Nombreuses (à la Lisp, inspirées d'APL)…

Alpha-extension (équivalent map)

`+^ {1, 2, 3} {4, 5, 6}` vaut `{5, 7, 9}`

Beta-réduction (équivalent fold)

`+\ {1, 2, 3}` vaut `6`

Balayage

`+\\ {1, 2, 3}` vaut `{1, 3, 5}`

Composition (équivalent concat)

`{1, 2}#{3, 4}` vaut `{1, 2, 3, 4}`

Sélection

`{4, 5}({1, 0, 1})` vaut `{5, 4, 5}`

Génération

`'n` vaut `{0, 1, 2, …, n-1}`

Troncature

`{1, 2}:[3]` vaut `{1, 2, 1}`

etc…

Exemples de programmes sur les collections

`C[g]` : spécifie explicitement la géométrie `g` de la collection `C`

Exemple 1 : iota (APL)

Soit n donné

```
iota[n] = 0 # (iota:[n-1] + 1)
```

Si n=9, alors définit la collection :

```
iota = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 }
```

Exemple 2 : Fibonacci

Soit n donné

```
fib[n] = 1 # fib:[n-1]
       + { 0, 0 } # fib:[n-2]
```

Exemple 3 : factorielle

Soit n donné

```
fact[n] = 1#((iota+1)*fact:[n-1])
```

Les tissus (*web*)

Définition

Tissu = flot de collections = collection de flots

Exemple : équation de diffusion

Problème : une barre de métal fine de longueur L, tenue à ses extrémités ; en tout point x de la barre, sa température à l'instant t est $u(x, t)$.

Question : étudier l'évolution de la température lorsque la barre est laissée à elle-même, connaissant la température à ses extrémités (x=0 ou x=L) et à t=0.

Solution : résoudre l'équation aux dérivées partielles suivante :

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}$$

Exploitation d'un schéma type différences finies :

- espace discrétisé : $x_i = i \,.\, h$ (h = pas spatial)
- temps discrétisé : $t_j = j \,.\, k$ (k = pas temporel)
- donc $u_{i,j} = u(x_i, t_j)$

$$\frac{u_{i,j+1} - u_{i,j}}{k} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2}$$

soit $u_{i,j+1} = ru_{i-1,j} + (1-2r)u_{i,j} + ru_{i+1,j}$ en posant $r = \frac{k}{h^2}$

En 8 ½

```
left[1]  = température en x=0, pour tout t
right[1] = température en x=L, pour tout t
u@0      = distribution température à t=0
u        = (left # V # right)
           when Clock

L        = ...     longueur de la barre
k        = ...     pas temporel
n        = 100     nombre total de points
m        = n-2     nombre total de points sauf extr.
h        = L/n
r        = k/(h*h)

ul       = ($U)('m)
um       = ($U)('m+1)
ur       = ($U)('m+2)

V[m]     = r*ul+(1-2*r)*um+r*ur
```

Premières leçons

Constat

Collection = agrégat homogène : trop contraignant

Solution : notion de système

Système = collection hétérogène

- les composants sont repérés par un nom

Exemple

```
s = 5,
Machine = {
  t@0      = 0,
  t        = $t+1 when Clock,
  pièce    = 0==(t%s)
}
```

ici `Machine` est un système composé de deux champs :

- `t` est un flot entier
- `pièce` est un flot booléen

Les flots sont accessibles à l'extérieur du système via une notation pointée :

- `Machine.t`
- `Machine.pièce`

Et alors ?

Permet de programmer avec un style objet, via un opérateur de concaténation des systèmes (#).

Exemple

```
Mobile = {
 position@0 = initial,
 position   = $position+déplacement
},

TrajectoireUniforme = Mobile # {
 déplacement = vitesse when Clock
}

soleil = TrajectoireUniforme # {
 vitesse = {1, 1},
 initial = {0, 0}
}
```

Les extensions à 8 ½ (1/2)

Les GBF

GBF = Group-Based Field

Généralise la notion de collection : champs de donnée

- agrégat homogène
- les index forment un groupe

un index est appelé forme

Propriétés du groupe des formes

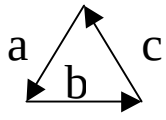
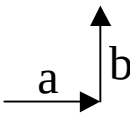
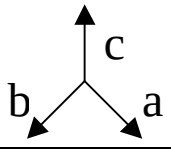
Le groupe (P, .) doit avoir une présentation finie (G, E)

- G = un ensemble fini de générateurs
- E = un nombre fini d'équations du type

$$w = w'$$

où w et w' sont des compositions par . d'éléments de G ou de leurs inverses

Exemples

Maillage triangulaire	Maillage carré	Maillage hexagonal
		
$a^2 = b^2 = c^2 = 0$ $(a.b.c)^2 = 0$	$a.b = b.a$	$b = a.c$

Propriétés du groupe des formes

Graphe de Cayley

Si (P, .) est le groupe des formes, alors il est possible de construire un graphe :

$$(P, A)$$

avec $A \subseteq P \times B$ tel que :

$$(x, y) \in A \text{ ssi il existe } z \text{ tel que } x.z = y.$$

Si B = l'ensemble des générateurs de la présentation finie (appelé aussi base), alors ce graphe est appelé graphe de Cayley du groupe.

Programmer avec des GBF

- si F est un GBF et P une forme (un point), alors

$$F(P) = f(F(P.g_1), F(P.g_2), \ldots, F(P.g_n))$$

où g_i = générateur du groupe, *f* fonction quelconque

- si F opère en tous les points de l'espace E, alors on note :

$$F[E] = f(F.g_1, F.g_2, \ldots, F.g_n)$$

Autrement dit :

+ le programme s'exprime sans référence absolue aux points

+ il s'exprime seulement en fonction de déplacements locaux (générateurs)

Les extensions à 8 ½ (2/2)

Les amalgames

Généralise (et systématise) la notion de système

- agrégat hétérogène

Principales opérations

Amalgamation

$A = \{ n_1 = e_1, \ldots, n_k = e_k \}$

Concaténation

$A = A_1 \# A_2$

Sélection

$A.n_i = e_i$

Le projet MGS

Constat

Tout programme 8½ (avec ou sans extension) se ramène au schéma de calcul suivant :

1. extraire d'une collection C une sous-collection A
2. calculer à partir de A une nouvelle sous-collection B
3. remplacer dans C la sous-collection A par B
4. revenir en 1.

Outillage théorique

La topologie algébrique

- définit formellement les notions de voisinage, de bord, de recollage (remplacement)
- généralise toutes les structures de données connues en informatique, y compris celles de 8½ (notion de complexe simplicial)

Définition d'un projet de recherche

MGS = un **M**odèle **G**énéral pour la **S**imulation

- tenter de caractériser topologiquement toutes les notions évoquées dans 8 ½
- en dériver de nouveaux concepts pour l'informatique

MGS en quelques mots

Reprise des idées de la ChAM

- dans une réaction chimique

  si x, y alors z

la notation x, y spécifie que les deux molécules sont distinctes, mais qu'en plus elles sont proches (au sens d'une topologie liée au problème)

$\Rightarrow$ structure de monoï de, avec l'opérateur ,

Les développements actuels

Un langage de programmation

- fonctionnel typé
- ses structures de données sont étudiées pour être traitées efficacement sur une machine massivement parallèle

Structures monoï dales développées :

- multi-ensembles

  opérateur , commutatif
- ensembles

  opérateur , commutatif et idempotent
- séquences

  opérateur , non commutatif

Conclusion

approche novatrice

- très en marge des écoles de pensée classiques (graphes, variantes λ-calcul, objets-agents, etc)
- outillage mathématique très élaboré

topologie algébrique

- semble un cadre général pour y plonger de nombreux concepts de l'informatique
- premiers résultats encourageants
  - + descriptions multi-échelle (à résolutions multiples)
  - + preuves de programmes parallèles
  - + nouvelles structures de données (GBF)