DEA informatique
université Montpellier II

Quels liens

entre

les problèmes de la modélisation – simulation de systèmes complexes

et

la programmation des machines massivement parallèles

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

9 décembre 2002

Plan

Introduction

Contexte de la modélisation – simulation

Questions à traiter :

- comment modéliser un système complexe
- comment exploiter un modèle en simulation (informatique)

Réponses de l'informatique :

- modèle exploité en simulation = programme
- la conception de programmes relève de la modélisation de systèmes compliqués, voire complexes
  $\Rightarrow$ méthodologies, outils d'aide à la modélisation
- les modèles ainsi obtenus sont directement exploitables en simulation.

Difficultés :

- quelle adéquation entre les propositions de l'informatique et les pratiques de la modélisation – simulation : compatibles ?
- de façon générale, la transformation d'un modèle (au sens large : un modèle = un document, au pire) en un programme n'est pas toujours facile.

Les apports de l'informatique à la modélisation : plusieurs facettes

L'approche représentation de connaissances

- représenter la connaissance
- représenter et opérationnaliser les raisonnements

Exemples :

- Ontologies : CYC, KIF, RDF(S)
- Représentations "naturelles" : réseaux sémantiques, graphes conceptuels

L'approche génie logiciel

- cycle de vie d'un programme, de sa conception à sa maintenance
- paradigmes de programmation

  impératif, objet, fonctionnel, contrainte (logique)

L'approche système d'information (BdD)

- modélisation des schémas de données, des traitements
- gestion de gros volumes de données

  distribution, concurrence, transactions

Approches alternatives

- paradigme multi-agent
- approches purement algébriques

Convergence de différentes approches : UML

- aboutissement des efforts des communautés objet (génie logiciel) et système d'information
- très prisé en ce moment (intégration dans la plupart des IDE)
- UML = langage orienté objet de modélisation
  ⇒ guide la conception d'un logiciel
  ⇒ les utilisateurs du futur logiciel auront des rôles parfaitement délimités
  ⇒ se veut un langage de communication (graphique) entre les acteurs humains du processus de conception
- contribution à la représentation des processus (diagrammes de séquences, d'états)
  ⇒ description d'une dynamique : pallie l'un des points faibles de l'approche objet
- de nombreux points restent à creuser

  + maîtriser l'évolution, la révision d'un modèle UML

  + exprimer et vérifier sa cohérence

  + dérivation automatique de code : faut-il ajouter à UML un langage pour décrire du code ?

  + rétro-conception UML à partir de code existant

Contexte des machines massivement parallèles

Question

- comment programmer une machine

  + composée d'un très grand nombre d'unités de calcul (PU)

  + ces PU sont connectées par un réseau dont la topologie de voisinage n'est pas nécessairement homogène

  + des PU peuvent être retirés au réseau en cours de calcul (PE non fiables, altérations du réseau)

  + des PU peuvent être ajoutés au réseau en cours de calcul

Réponse

- il n’y en a pas d’aboutie à ce jour

  + plusieurs voies de recherche ouvertes

  + aucune n'a pris le pas sur les autres

  ⇒ bref, pour l’instant, tout le monde patauge ;-)

Quelques pistes (liste non exhaustive)

Ignorer la dimension parallèle

Faire abstraction du parallélisme de la machine

- Haskell

  rester dans un cadre de programmation classique (ici cadre fonctionnel) : le compilateur est chargé d'exploiter au mieux les ressources de calcul

Ignorer les contraintes matérielles

La machine est parallèle, mais abstraction de son architecture

- PVM (Parallel Virtual Machine)

  programmer = définir un ensemble de processus communicants, sans mémoire commune, qui échangent des données via des messages

- MPI (Message Passing Interface)

  programmer = gérer des échanges de messages (pas de notion de processus explicite)

Tenir compte de tout

La machine est parallèle, et son architecture est prise en compte dans le code

- Occam

  programmer = recevoir et émettre des données sur les canaux de communication associés à chaque unité de calcul (Transputer).

Trouver les bonnes abstractions du parallélisme

- langage 8.5

  programmer = définir des équations sur des tissus. Un tissu est une collection (agrégat spatial) de flots (agrégats temporels)

- MGS, une extension de 8.5

  programmer = définir des équations sur des objets algébriques

  ⇒ des opérations algébriques dénotent des relations spatio-temporelles.

Penser un programme comme la maîtrise d'un processus physique

Fondamentalement, du point de vue de la Physique

- un programme est un champ opérant sur la mémoire
- exécuter un programme, c'est dérouler la trajectoire d'un point (état mémoire) soumis à ce champ

  \+ donnée d'entrée = début de la trajectoire

  \+ résultat = point fixe de la trajectoire (s'il existe)

Quelques métaphores constructives

- métaphore chimique (L.M. Adleman)

  \+ la machine chimique abstraite de G. Berry et G. Boudol

- métaphore biologique (cellulaire)

  \+ les systèmes membranaires de G. Paun

  \+ la machine autodéveloppante de F. Gruau

⇒ objectif : repenser en profondeur la programmation des machines, et donc notre façon de percevoir (modéliser) le monde…

La machine chimique abstraite

The CHemical Abstract Machine (CHAM) : publié en 1990, par G.Berry et G. Boudol.

Présentation

- Un état instantané de la machine est une **solution**, i.e. un multi-ensemble de **molécules**

$$S = \{| m_1, m_2, ..., m_k |\}$$

- Une molécule est un terme algébrique (arbre par exemple).
- Une **règle de réaction** consomme des molécules pour en produire d'autres

  + Règles structurales (réversibles)

    $S \leftrightarrow S'$ (où $\rightarrow$ = échauffement, $\leftarrow$ = refroidissement)

  + Règles de réduction (irréversibles)

    $S \rightarrow S'$

Sémantique

soient S, S' et S'' trois solutions, et C[T] un contexte, i.e. une molécule paramétrée par un terme T.

- loi chimique :

  si $S \Rightarrow S'$, alors $S \oplus S'' \Rightarrow S' \oplus S''$

- loi des membranes :

  si $S \Rightarrow S'$, alors $\{| C[S] |\} \Rightarrow \{| C[S'] |\}$

Résultats

C’est un modèle de calcul universel

Rappelle furieusement les bons vieux systèmes à règles de production des années 80

Formalisation naturelle en logique linéaire

(cf J.Y. Girard ou Y. Lafont)

Réalisation

difficulté particulière :

- il faut garantir que si une réaction est possible, elle aura bien lieu

$\Rightarrow$ c’est le point faible de toutes les tentatives de réalisation existantes

- technique de compilation des règles type *algorithme de RETE*

Le langage Gamma

GAMMA = General Abstract Model for Multiset mAnipulation.

Publié par J.P. Banâtre et D. Le Metayer (1988).

Ce travail a fortement inspiré les auteurs de la CHAM.

Exemples

max : **replace** x, y **by** y **when** x<= y

sort : **replace** (i, x), (j, y) **by** (i, y), (j, x)
when i > j and x <y

iota : **replace** (x, y) **by** (x, (x+y)/2), ((x+y)/2+1, y)
when x < y
replace (x, y) **by** x **when** x = y

sieve : **replace** x, y **by** y **when** x%y==0

primes(n) = sieve(iota((2, n)))

Extensions

- sur les types : Structured Gamma
- sur la notion de membrane : Higher-Order Gamma

Les systèmes de Paun

Pour résumer

Sorte de machine chimique :

- **molécules** avec **règles de réaction**
- regroupement de molécules via des **membranes**, sachant que :

  + les membranes sont imperméables (une molécule ne peut pas changer de membrane)

  + une membrane peut être dissoute par une molécule particulière (réaction spéciale)

  + il existe une membrane encapsulant toutes les autres, qui ne peut être dissoute

Sémantique

A l'initialisation, un système membranaire est donné :

- une hiérarchie de membranes (toute intersection de deux membranes A et B est soit vide, soit égale à A ou B $\Rightarrow$ le graphe d’inclusion est un arbre)
- des molécules sont placées dans chaque membrane

Ensuite, les règles de réaction sont appliquées jusqu'à saturation (plus de réaction).

La machine autodéveloppante de Gruau

Travaux menés essentiellement par F. Gruau.

Pour résumer

Machine (virtuelle) dont le nombre d'unités de calcul et leurs connexions sont modifiables en cours de calcul et contrôlées par le programme.

Description

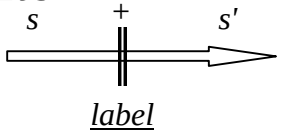
- Le graphe de calcul
  + des noeuds ports (en nombre fixe) et des noeuds de calcul
  + des arcs orientés
- L'automate des transitions d'état
  + une fonction de transition d'état
  + une fonction de production d'un signal = une instruction

Pour chaque noeud du graphe de calcul :

- son état

Pour chaque arc du graphe de calcul :

- son signal à gauche et son signal à droite
- sa porte (ouverte = ou fermée ||)
- sa polarité (+ ou -)
- son étiquette



Sémantique

Automate des transitions d’état

- fonction de transition d’état

  selon

  + l’état courant

  + l’ensemble des signaux des arcs attenants

  alors

  + précise l’état suivant

  + produit un symbole :

  - soit il s’agit d’une instruction pour le noeud

  - soit il s’agit d’un signal (peut être une instruction d’arc) pour tout arc attenant étiqueté e

Mise à jour du graphe de calcul

- phase 1 : pour chaque noeud de calcul

  a) si le noeud est isolé, le supprimer ; sinon, change son état conformément à la fonction de transition

  b) si le symbole produit par l’automate est une instruction de noeud, alors elle est exécutée, sinon tous les arcs concernés sont mis à jour

- phase 2 : pour chaque arc

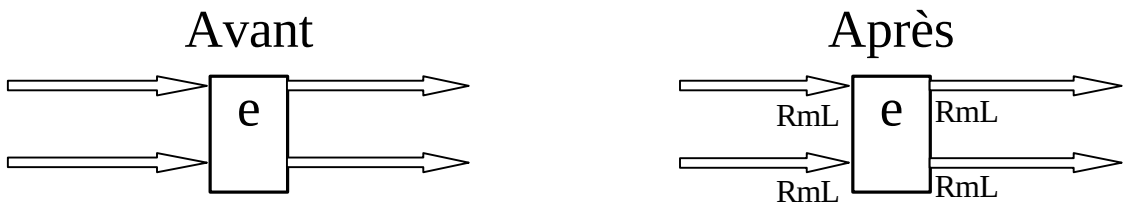
  a) si l’un des deux signaux est une instruction d’arc, l’exécuter (si deux instructions, respecter les priorités), et remplacer ce signal par le signal spécial Nop.

  b) échanger les deux signaux si la porte est ouverte, sinon rien

Les signaux instruction en image
- instructions structurelles de noeud -

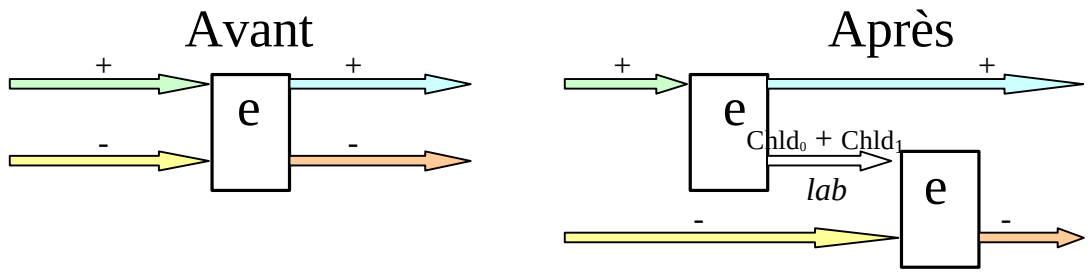
L'instruction RmN (remove node)

Symbole à produire par l'automate : RmN



L'instruction CpN (copy node)

Symbole à produire par l'automate : (CpN, *lab*)



L'instruction Mrg (merge node)

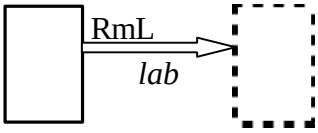
Symbole à produire par l'automate : (Mrg, *lab*, x)

Les signaux instruction en image
- instructions structurelles d’arc -

L'instruction d'arc RmL (remove link)

Symbole à produire par l'automate : (RmL, *lab*)

Avant

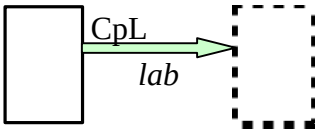


Après

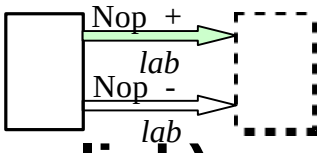
L'instruction d'arc CpL (copy link)

Symbole à produire par l'automate : (CpL, *lab*)

Avant



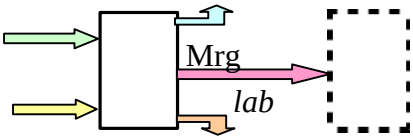
Après



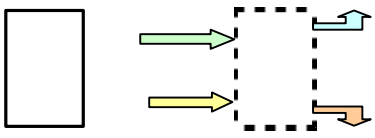
L'instruction d'arc Mrg (merge link)

Symbole à produire par l'automate : Mrg

Avant



Après



Autres signaux instruction

- Instructions d'arc
  + positionner la polarité
  + positionner la porte
- Instructions système
  + Nop : ne rien faire
  + $Chld_0$ et $Chld_1$: signaux de création d'un noeud
  + Sync : synchronisation
- Sémantique des noeuds ports

  Noeud port = noeud connecté au monde extérieur via un canal d'entrée et un canal de sortie

  Lors de la phase de mise à jour des noeuds (phase 1) :

  - le signal écrit sur l'arc incident est envoyé au monde extérieur via le canal de sortie (out)
  - le signal provenant du monde extérieur (canal d'entrée) est écrit sur l'arc incident (in)

Fonctionnement global

Configuration initiale

Le graphe de calcul initial est constitué de :

- *k* noeuds ports
- un seul noeud de calcul *N*, d'état q_0 (initial)
- les seuls arcs sont ceux allant de N vers chaque port

  chaque arc est polarisé +, porte ouverte, étiqueté différemment de tous les autres.

Un exemple

une pile

Points non exposés

La synchronisation

La machine autodéveloppante – résultats -

- Modèle de calcul universel
- Parallélisme massif

  La bipartition du graphe assure que :

  + en phase 1, la mise à jour des noeuds peut être réalisée en //, et est confluante

  + en phase 2, idem pour la mise à jour des arcs
- Développe sa puissance de calcul en fonction des besoins

Questions

- Quel langage de programmation est adapté à ce type de machine

  Quelles structures de données

  Quelles structures de contrôle
- Quelle architecture matérielle dédiée pourrait se prêter à son mode de fonctionnement

Autres modèles de calcul exotiques

Les modèles réalistes

Les ordinateurs amorphes (G.J. Sussman)

Présentés comme une extension des automates cellulaires (éliminations de contraintes)

- Elément de base = une unité de calcul (UC) ayant

  deux points de connexion

  peu coûteuse à fabriquer

  de très petite taille

  une puissance de calcul limitée

  même programme pour toutes les UC

- Machine = un volume dans lequel est injecté en vrac un très grand nombre d'UC

  Principe : quand deux UC ont un point de connexion proche, les deux points de connexion fusionnent $\Rightarrow$ UC reliées

  Le réseau a donc une topologie non prévisible.

- Sémantique

  A chaque instant, une UC n'échange qu'avec une UC voisine à la fois

  Le choix de l'UC voisine est non déterministe

Les modèles papiers

Modèle pour lequel personne ne sait s'il est réalisable

- les machines quantiques

Conclusion

1. La simulation de systèmes complexes implique de fortes puissances de calcul
2. Seules des machines massivement parallèles peuvent soutenir une telle puissance
3. Personne ne sait aujourd'hui exploiter pleinement ces machines

Tout modèle numérisé devient une véritable maquette

Le comportement du monde simulé par la machine vise à être en adéquation avec le comportement observé du monde réel modélisé

Autrement dit…

- Notre façon de modéliser le monde nous oblige à changer notre façon de programmer les machines,
- Qui à son tour nous obligera peut-être à changer notre façon de modéliser…

Cf. les avancées en topologie algébrique, avec en particulier des applications prometteuses :

+ en informatique (preuve de programmes parallèles, nouvelles structures de données)

+ en physique fondamentale (super-cordes)