Cours 8

Objets - Classes - Méthodes

Jean-Jacques.Levy@inria.fr

http://jeanjacqueslevy.net

secrétariat de l'enseignement:

Catherine Bensoussan

cb@lix.polytechnique.fr

Laboratoire d'Informatique de l'X

Aile 00, LIX

tel: 34 67

http://w3.edu.polytechnique.fr/informatique

Références

- A Theory of Objects, Martín Abadi, Luca Cardelli, Springer, 1996.
  http://www.luca.demon.co.uk/TheoryOfObjects.html
- Foundations for Programming Languages, J. Mitchell, MIT Press, 1996.
- Semantics of Programming Languages, C. Gunter, MIT Press, 1992.
- Programming Languages, Concepts and Constructs, Ravi Sethi, 2nd edition, 1997.
  http://cm.bell-labs.com/who/ravi/teddy/
- Theories of Programming Languages, John C. Reynolds, Cambridge University Press, 1998.
  http://www.cs.cmu.edu/afs/cs.cmu.edu/user/jcr/www/

Plan

1. Un calcul des objets
2. Exemples
3. Objets modifiables
4. Typage des objets $\neq$ sous-typage
5. Les sujets non traités
6. DEA $+$ recherche

Les objets en PCF [Abadi-Cardelli 96]

Termes

$$
\begin{array}{rcll}
M, N, P & ::= & \ldots & \textbf{voir cours précédents} \\
& \mid & \{l_1 = \zeta x_1.M_1; l_2 = \zeta x_2.M_2; \ldots l_n = \zeta x_n.M_n\} & \textbf{objet } (n \geq 0) \\
& \mid & M.l & \textbf{valeur d'un champ} \\
& \mid & M.l \Leftarrow \zeta x.N & \textbf{modification d'un champ}
\end{array}
$$

On écrira aussi $\{l_i = \zeta x_i.M_i{}^{i \in 1..n}\}$ pour un objet. (l_i distincts)
$\zeta x.M$ est une méthode.

Valeurs

$$
\begin{array}{rcll}
V & ::= & \ldots & \textbf{comme avant} \\
& \mid & \{l_i = \zeta x_i.M_i{}^{i \in 1..n}\} &
\end{array}
$$

Variables libres

ζ est un nouveau lieur. La variable liée parfois appelée *self* ou *this*.

$$
\begin{array}{ll}
FV(M.l) = FV(M) & FV(\{l_i = \zeta x_i.M_i{}^{i \in 1..n}\}) = \bigcup_{i=1}^{n} FV(\zeta x_i.M_i) \\
FV(\zeta x.M) = FV(M) - \{x\} & FV(M.l \Leftarrow \zeta x.N) = FV(M) \cup FV(\zeta x.M)
\end{array}
$$

Réductions

Soit $o = \{l_i = \varsigma x_i.M_i{}^{i \in 1..n}\}$

m_invocation	$o.l_i \to M_i[x_i \backslash o]$
m_update	$o.l_i \Leftarrow \varsigma x.M \to \{l_i = \varsigma x.M;\ l_j = \varsigma x_j.M_j{}^{j \in (1..n)-i}\}$

Abréviation pour les champs données

Si $\varsigma x.M$ **est une donnée (**$x \notin FV(M)$**), on pose**

$$\begin{array}{rcl} \{\ldots; l = M; \ldots\} & = & \{\ldots; l = \varsigma x.M; \ldots\} \\ o.l \Leftarrow M & = & o.l \Leftarrow \varsigma x.M \end{array}$$

Alors on a bien $(o.l \Leftarrow M).l \to M$**.**

Exemples

Si $P = \{x = \underline{3}; y = \underline{4}\}$
$P.x \to \underline{3} \quad P.y \to \underline{4}$
$P.x \Leftarrow \underline{5} \to \{x = \underline{5}; y = \underline{4}\} \quad P.x \Leftarrow \underline{5}.\ y \Leftarrow \underline{6} \to \{x = \underline{5}; y = \underline{6}\}$

Exemples

Si $P = \{x = \underline{3}; y = \underline{4};$

$$move\_x = \zeta o.\lambda d.\, o.x \Leftarrow o.x + d;$$

$$move\_y = \zeta o.\lambda d.\, o.y \Leftarrow o.y + d\}$$

$P.move\_x\ \underline{2}$

$\rightarrow (\lambda d.\, P.x \Leftarrow P.x + d)\underline{2}$

$\rightarrow P.x \Leftarrow P.x + \underline{2}$

$\rightarrow P.x \Leftarrow \underline{3} + \underline{2}$

$\rightarrow P.x \Leftarrow \underline{5}$

$\rightarrow \{x = \underline{5}; y = \underline{4};$

$\quad move\_x = \zeta o.\lambda d.\, o.x \Leftarrow o.x + d;$

$\quad move\_y = \zeta o.\lambda d.\, o.y \Leftarrow o.y + d\}$

De même,

$(P.move\_x\ \underline{2}).move\_y\ \underline{3}$

$\rightarrow^* \{x = \underline{5}; y = \underline{7};$

$\quad move\_x = \zeta o.\lambda d.\, o.x \Leftarrow o.x + d;$

$\quad move\_y = \zeta o.\lambda d.\, o.y \Leftarrow o.y + d\}$

Exemple des cellules mémoires

$$
\begin{array}{l}
\texttt{let } c = \{\, \mathit{contents} = \underline{0};\\
\qquad\qquad \mathit{get} = \zeta s.\, s.\mathit{contents};\\
\qquad\qquad \mathit{set} = \zeta s.\, \lambda v. s.\mathit{contents} \Leftarrow v\}\\
\texttt{in } ((c.\mathit{set}\ \underline{3}).\mathit{set}\ (c.\mathit{get} + \underline{1})).\mathit{get}\\
\to^* \underline{4}
\end{array}
$$

Une cellule mémoire avec sauvegarde.

$$
\begin{array}{l}
\texttt{let } c = \{\, \mathit{contents} = \underline{0};\\
\qquad\qquad \mathit{get} = \zeta s.\, s.\mathit{contents};\\
\qquad\qquad \mathit{set} = \zeta s.\, \lambda v.\, (s.\mathit{backup} \Leftarrow s.\mathit{contents}).\mathit{contents} \Leftarrow v;\\
\qquad\qquad \mathit{backup} = \underline{0};\\
\qquad\qquad \mathit{restore} = \zeta s.\, s.\mathit{contents} \Leftarrow \mathit{backup}\}\\
\texttt{in } \ldots
\end{array}
$$

La même en utilisant deux lieurs *self* différents.

$$
\begin{array}{l}
\texttt{let } c = \{\, \mathit{contents} = \underline{0};\\
\qquad\qquad \mathit{get} = \zeta s.\, s.\mathit{contents};\\
\qquad\qquad \mathit{set} = \zeta s.\, \lambda v.\, (s.\mathit{restore} \Leftarrow \zeta t.\, t.\mathit{contents} \Leftarrow s.\mathit{contents}).\mathit{contents} \Leftarrow v;\\
\qquad\qquad \mathit{restore} = \zeta s.\, s.\mathit{contents} \Leftarrow \underline{0}\}\\
\texttt{in } \ldots
\end{array}
$$

Calcul des objets purs

Exercice Soit
$$\begin{aligned} o_1 &= \{l = \zeta x.\{\,\}\} \\ o_2 &= \{l = \zeta x.\, x.l\} \\ o_3 &= \{l = \zeta x.\, x\} \\ o_4 &= \{l = \zeta y.\, y.l \Leftarrow \zeta x.x\} \end{aligned}$$
Montrer que $o_1.l \to \{\,\}$, $o_2.l \to o_2.l$, $o_3.l \to o_3$, $o_4.l \to^* o_3$

Récursion

Posons $\mu x.M = \{mu = \zeta x.\, M[x\backslash x.mu]\}.mu$ Alors on a

$$\begin{array}{lcl} \mu x.M & \to & M[x\backslash x.mu][x\backslash\{mu = \zeta x.\, M[x\backslash x.mu]\}] \\ & = & M[x\backslash\{mu = \zeta x.\, M[x\backslash x.mu]\}.mu] \\ & = & M[x\backslash \mu x.M] \end{array}$$

Exercice Montrer qu'on peut coder tout le lambda calcul avec les seuls objets.

Classes et sous-classes

On suppose donné un ensemble de fonctions $\lambda s.M_i$, comment construire une classe à partir d'elles?

Une classe sera:

$$\begin{aligned} c = \{\, & new = \zeta z.\,\{l_i = \zeta s.(z.l_i)(s)\ ^{i\in 1..n}\};\\ & l_i = \lambda s.M_i{}^{i\in 1..n}\} \end{aligned}$$

On crée un objet o par $o = c.new \to \{l_i = \zeta x_i.M_i{}^{i\in 1..n}\}$

Pour créer une sous-classe avec des fonctions supplémentaires $\lambda s.M_k$, on écrit

$$\begin{aligned} c' = \{\, & new = \zeta z.\,\{l_i = \zeta s.(z.l_i)(s)\ ^{i\in 1..n+m}\};\\ & l_j = c.l_j{}^{j\in 1..n};\\ & l_k = \lambda s.M_k{}^{i\in n+1..n+m}\} \end{aligned}$$

Quelques remarques

- les objets ne sont pas modifiables
- les opérations génèrent de nouveaux objets
- le calcul est complètement fonctionnel

Objets modifiables

Termes

$$\begin{array}{lcll} M, N, P & ::= & \ldots & \textbf{comme avant} \\ & | & clone(M) & \textbf{clonage} \end{array}$$

Une location peut aussi être une valeur.

Règles de réduction

alloc $\langle \{l_i = \zeta x_i.M_i{}^{i \in 1..n}\},\ s\rangle \to \langle \ell,\ s + [\ell = \{l_i = \zeta x_i.M_i{}^{i \in 1..n}\}]\rangle$

$(\ell \notin \mathit{domaine}(s))$

m_clone $\langle clone(\ell),\ s\rangle \to \langle \ell',\ s + [\ell' = s(\ell)]\rangle$ $(\ell' \notin \mathit{domaine}(s))$

Si $s(\ell) = \{l_i = \zeta x_i.M_i{}^{i \in 1..n}\}$

m_invoc $\langle \ell.l_i,\ s\rangle \to \langle M_i[x_i \backslash \ell],\ s\rangle$

m_upd $\langle \ell.l_i \Leftarrow \zeta x.M,\ s\rangle \to \langle \ell,\ s[\ell \backslash \{l_i = \zeta x.M;\ l_j = \zeta x_j.M_j{}^{j \in (1..n)-i}\}]\rangle$

En fait, $clone$ **est une opération dérivée.**

Typage des objets

Si un objet o de type t a au moins tous les champs de o' de type t', il est logique de dire que t est un sous-type de t', noté $t <: t'$.

Par exemple, la cellule mémoire avec $backup$ a un sous-type du type général des cellules (cf. exemples plus haut).

$$\begin{aligned} P : t = \{\, & x = \underline{3};\ y = \underline{4}; \\ & move\_x = \zeta o.\lambda d.\, o.x \Leftarrow o.x + d; \\ & move\_y = \zeta o.\lambda d.\, o.y \Leftarrow o.y + d\} \end{aligned}$$

$$\begin{aligned} P_c : t_c = P \texttt{ with } \{c = rouge\} = \{\, & x = \underline{3};\ y = \underline{4};\ c = rouge; \\ & move\_x = \zeta o.\lambda d.\, o.x \Leftarrow o.x + d; \\ & move\_y = \zeta o.\lambda d.\, o.y \Leftarrow o.y + d\} \end{aligned}$$

Alors $t_c <: t$
(Les points colorés forment un sous-ensemble des points).

Quel est le type de $move\_x$? de $move\_y$?

Le sous-typage ne suffit pas à typer les objets [Cook, Hill, Canning 90]

La méthode $\zeta o.\lambda d.\, o.x \Leftarrow o.x + d$ prend un point et un entier pour retourner un point.

$move\_x : t \to \mathtt{int} \to t$

Comme $t_c <: t$, on peut aussi l'appliquer à un point coloré. On peut donc garantir que

$P_c.move\_x\ 3 : t$

Mais peut-on dire que $P_c.move\_x\ 3 : t_c$?

Si on répond oui, alors le code de $move\_x$ doit recopier tout les champs, couleur comprise, sinon il y a risque d'erreur à l'exécution. Donc son code ne dépend pas uniquement de son type!

Problème avec les types

Plus grave si on rajoute la méthode $equals$ retournant $\underline{1}$ si les deux points sont égaux, et $\underline{0}$ sinon (en l'absence de booléens en PCF).

$$
\begin{aligned}
P : t = \{ & x = \underline{3};\ y = \underline{4}; \\
& \quad equals = \zeta o.\lambda o'.\,(o.x = o'.x) \otimes (o.y = o'.y)\}
\end{aligned}
$$

$$
\begin{aligned}
P_c : t_c \ =\ & P \ \texttt{with}\ \{ c = rouge; \\
& \qquad equals = \zeta o.\lambda o'.\,(o.x = o'.x) \otimes (o.y = o'.y) \otimes (o.c = o'.c)\} \\
=\ & \{ x = \underline{3};\ y = \underline{4};\ c = rouge; \\
& \quad equals = \zeta o.\lambda o'.\,(o.x = o'.x) \otimes (o.y = o'.y) \otimes (o.c = o'.c)\}
\end{aligned}
$$

Soient deux points P, P' et deux points colorés P_c, P'_c.
$P.equals\ P'$ et $P_c.equals\ P'_c$ sont bien typés.

Mais comme $t_c <: t$, le terme $P_c.equals\ P'$ est aussi bien typé.
Or il provoque une erreur à l'exécution.

Typage des objets et sous-typage sont différents

Covariance / Contravariance

Le type d'une fonction est contravariant sur son argument et covariant dans son résultat. I.e

$$t <: t' \quad u <: u' \quad \Rightarrow \quad t' \rightarrow u <: t \rightarrow u'$$

Le type des objets et des méthodes doit en tenir compte.

Une autre solution est de ne pas autoriser de sous-typage ou plutôt de le rendre explicite, c'est la solution d'Ocaml. La relation $<:$ apparait explicitement dans les programmes. L'avantage est de permettre l'inférence de types.

En Eiffel, on se sert d'informations dynamiques. En C++, Modula-3 et Java, le type des méthodes spécialisées ne peut changer.

Le typage des objets: problème difficile

- le typage des objets est un problème toujours ouvert.
- peut-on faire inférence et types sophistiqués?
- types des objets concurrents? (*RMI*, *Network objects*)
- Ocaml privilégie l'inférence
- Java est une simplification de C++, qui profite de la technologie des langages de programmation.
- Conférences sur les objets: OOPSLA, ECOOP, POPL

Les problèmes non traités dans le cours

- les types des objets
- la surcharge (cf. Haskell ou Java ou C++)
- les modules (paramétriques)
- les exceptions (dans les types?)
- les types récursifs
- l'équivalence de types

- les démonstrations des théorèmes

Pour en savoir plus Aller en DEA, en thèse.
cours de Leroy, Rémy, Castagna, Pottier au DEA SPP

La recherche

- l'informatique est un vaste domaine scientifique
- l'informatique est en plein boum économique, scientifique
- théorie et pratique ne sont pas éloignées
- en informatique, le doctorat = diplôme connu à l'étranger
- France = pensée formelle ⇒ informatique++
- logiciel de base = triple bonus
- recherche intéressante = recherche à cheval entre théorie et pratique.
- cursus de l'X ne favorise pas la recherche. Venez-nous voir pour plus de détails.
- France/Europe forte en langages de programmation. Ada, Prolog, ML, Modula, Pascal, Simula, Estérel, Lustre. Amérique: Lisp, C, Java.

La recherche – suite

- un domaine se mesure au nombre de ses conférences annuelles de bonne tenue. Exemple pour langages de programmation: POPL, ICFP, PLDI, OOPSLA, ECOOP, PPDP, LICS, PEPM, PPOPP. http://www.acm.org/sigplan/
- la recherche se mesure aussi au nombre de personnes déçues par un domaine (très peu en informatique)
- la seule bonne recherche est internationale
- être ambitieux (s'attaquer à des problèmes ouverts ou avec concurrence)
- les ex-élèves de la majeure se débrouillent bien en général

2 exemples de DEA: DEA Algorithmique

1. Algorithmique et combinatoire des mots, [J. Berstel]
2. Géométrie computationnelle, [J.-D. Boissonnat]
3. Introduction au calcul parallèle et distribué, [M. Morvan]
4. Algorithmes probabilistes, [J.-M. Steyaert]

. Pratique du calcul formel.

Filières, responsables et cours

1. Analyse d'algorithmes [J.-M. Steyaert]:
2. Automates et mots [J.-E. Pin]:
3. Calcul formel [D. Lazard]:
4. Combinatoire [R. Cori]:
5. Complexité, codage et cryptographie [J. Stern]:
6. Géométrie, images et robotique [O. Faugeras]:
7. Parallélisme et concurrence [A. Petit]:

. Conception de circuits, [P. Bertin, J. Vuillemin].

http://w3.edu.polytechnique.fr/informatique/

DEA Sémantique, Preuves et Programmation

1. Les termes en logique du premier ordre, [J.P. Jouannaud]
2. Lambda calcul, [T. Hardin]
3. Preuves constructives, [G. Dowek]
4. Sémantique dénotationnelle, [R. Di Cosmo]
5. Typage et programmation, [M. Mauny - D. Rémy - X. Leroy]

Filières, responsables et cours

1. Langages, [G. Cousineau] Langages distribués, [C. Queinnec]; Sous-typage et langages à objets, [G. Castagna - F. Rouaix - D. Rémy] Compilation des langages fonctionnels et impératifs, [X. Leroy] Programmation logique avec contraintes et systèmes concurrents, [F. Fages - L.Fribourg]

2. Modèles Sémantiques, [P.-L. Curien] Types, catégories et domaines, [P.-L. Curien-G. Longo] Logique linéaire, [R. Di Cosmo] Concurrence et Communication, [G. Gonthier - J.-J. Lévy] Modèles algébriques des processus et méthodes de vérification, [Ph. Schnoebelen]

3. Preuves et spécifications, [J.-P. Jouannaud] Calcul des constructions inductives, [C. Paulin - B. Werner] La méthode B, [V. Donzeau-Gouge, M. Simonot] Preuve par des techniques d'automates, [H. Comon] Réécriture et preuves, [J.-P. Jouannaud - C. Marche - M. Rusinowitch]

4. Sémantique et Interprétation abstraite, [R. Cousot] Fondements de l'interprétation abstraite, [P. Cousot] Interprétation abstraite: applications, [A. Deutsch - P. Granger - R. Cridlig] Langages objets, contraintes et typage, [F. Bourdoncle - B. Monsuez] Parallélisme : sémantique et preuve, [R. Cousot - E. Goubault - I. Mackie]

`http://w3.edu.polytechnique.fr/informatique/`

Laboratoires de recherche/Hitech

CNET Issy, Lannion
CNRS
INRIA, 5 Unités de recherche
ENS
ENS Cachan
ENS Lyon
ENPC
ENSMP
ENST
LIX

Labri Bordeaux
Imag Grenoble
Marseille Luminy
Paris 6
Paris 7, (Logique)
Marne la Vallée
ORSAY
Saint Denis
CNAM

Thomson LCR
Alcatel Marcoussis
GENSET
ILOG
O2 Technology
Chorus Systèmes
Xerox Grenoble
Web Consortium
Trusted Logic
Cryo Networks
Intel

Lucent
AT&T Bell labs
Bellcore
Compaq SRC, WRL
IBM Almaden, Yorktown
Microsoft Research Cambridge
Microsoft Research Redmond
Xerox PARC
HPlabs
SUN Microsystems

Domaines de recherche (exemple de l'INRIA)

Thèmes

1. Réseaux et systèmes
2. Génie logiciel et calcul symbolique
3. Interaction homme-machine, images, données, connaissances
4. Simulation et optimisation de systèmes complexes

Action de développements

DYADE : conception de systèmes d'information avancés
GÉNIE : science de l'information et ingénierie concourante
MÉDIACULTURE : système multimédia distribué pour la documentation culturelle
PRAXITELE : transport urbain public individuel
PRÉVISIA : interopérabilité des systèmes d'information
SYNCHRONE : environnement de développement de systèmes temps-réel
W3C : les services d'information sur Internet
WEBTOOLS : outils logiciels pour le World Wide Web

http://www.inria.fr/Recherche/activites-fra.html

Thème 1

Parallélisme et architecture :

APACHE - Algorithmique parallèle et partage de charge
API - Architectures parallèles intégrées
CAPS - Compilation, architectures parallèles et systèmes
PAMPA - Environnement de programmation des architectures massivement parall èles
ReMaP - Régularité et parallélisme massif
SLOOP - Simulation, langages orientés objets et parallélisme

Réseaux, systèmes, évaluation de performances :

MÉVAL - Modélisation et évaluation des systèmes informatiques
MISTRAL - Modélisation en informatique et systèmes de télécommuni cation : recherche et applications logicielles
MODEL - Modélisation de systèmes aléatoires
RESEDAS - Outils logiciels pour les télécommunications et les systèmes d istribués
RODEO - Réseaux à haut débit, réseaux ouverts
SATURNE - Systèmes répartis tolérant les fautes et les intrusions
SIRAC - Systèmes informatiques répartis pour applications coopératives
SOLIDOR - Architectures et systèmes distribués extensibles, tolérance au x fautes, programmation objet
SOR - Systèmes d'objets répartis

Programmation distribuée et temps-réel :

ADP - Algorithmes répartis et protocoles
EPM-PATR - Environnement de programmation pour applications temps réel
MEIJE - Parallélisme, synchronisation et temps réel
PARA - Parallélisme, mobilité
REFLECS - Systèmes informatiques répartis temps réel tolérant les fautes
SPECTRE - Spécification et programmation des systèmes communicants et tem ps réel

Thème 2

Sémantique et programmation :

CALLIGRAMME - Logique linéaire, réseaux de démonstration et grammaires cat égorielles
COQ - Spécifications et preuves de programmes
CRISTAL - Programmation typée, modularité et compilation
CROAP - Conception et réalisation d'outils d'aide a' la programmation
EURECA - Preuve, calcul symbolique et logique
LANDE - Langages déclaratifs
LOCO - Programmation logique avec contraintes
OSCAR - Outils syntaxiques pour la construction et l'analyse de programmes
PROTHEO - Contraintes, déduction automatique et preuves de propriétés de logiciels

Algorithmique et calcul formel :

ALGO - Algorithmes
CODES - Codes et protection de l'information
PRISME - Géométrie, algorithmes et robotique
SAFIR - Systèmes algébriques formels pour l'industrie et la recherche

Thème 3

Bases de données, bases de connaissances, systèmes cognitifs :

ACACIA - Acquisition des connaissances pour l'assistance a' la conception par interaction entre agents
AIRELLE - Représentations et langages
ATOLL - Atelier d'outils logiciels pour le langage naturel
DIALOGUE - Dialogue homme-machine a' forte composante langagière
OPERA - Outils pour les documents électroniques : recherche et applications
ORION - Modélisation des connaissances pour l'automatisation des tâches
PSYCHO-ERGO - Psychologie ergonomique pour l'informatique
REPCO - Représentation des connaissances
RODIN - Systèmes de bases de données
SHERPA - Bases de connaissances a' objets
SHOOD - Méthodes et outils pour l'intégration de systèmes industriels
SYCO - Systèmes de compréhension et bases de connaissances
VERSO - Bases de données

Vision, analyse et synthèse d'images :

AIR - Traitement d'image et données satellites dynamiques
EPIDAURE - Imagerie et robotique médicale
iMAGIS - Modèles, algorithmes, géométrie pour le graphique et l'image de synthèse
ISA - Image, synthèse, analyse
MOVI - Modélisation, localisation, reconnaissance et interprétation en vis ion parordinateur
PASTIS - Analyse de scènes et traitement d'images symboliques
ROBOTVIS - Vision par ordinateur et robotique
SHARP - Programmation automatique et systèmes décisionnels en robotique
SIAMES - Synthèse d'image, animation, modélisation et simulation
SYNTIM - Analyse et synthèse d'images
TEMIS - Traitement, exploitation et modélisation d'images séquentielles

Thème 4

Automatique, robotique, signal :

AS - Automatique et signal
ATGC - Action transversale génome et calcul
BIP - Conception et contrôle de robots marcheurs et applications
COMORE - Contrôle et modélisation de ressources renouvelables
CONGÉ - Contrôle géométrique des systèmes non linéaires
FRACTALES - Approche fractale pour l'analyse et la modélisation de signaux complexes
ICARE - Instrumentation, commande et architecture des robots évolués
MÉTA 2 - Méta automatique et méthodes de l'automatique
MIAOU - Mathématiques et informatique de l'automatique et de l'optimisation pour l'utilisateur
PROMATH - Programmation mathématique
SAGEP - Simulation, analyse et gestion des systèmes de production
SOSSO - Applications et outils de l'automatique
SYSTOL - Modélisation statistique et applications biomédicales

Modélisation et calcul scientifique :

ALADIN - Algorithmes adaptés au calcul numérique intensif
CAIMAN - Calcul scientifique, modélisation et analyse numérique
ESTIME - Estimation de paramètres
IDOPT - Identification et optimisation de systèmes en physique et en environnement
M3N - Multi-modèles et méthodes numériques
MODULEF - Méthodes et outils pour le calcul scientifique
NUMATH - Analyse mathématique et traitement numérique de modèles non linéaires
OMEGA - Méthodes numériques probabilistes
ONDES - Modélisation, analyse, simulation des équations des ondes
SINUS - Simulation numérique dans les sciences de l'ingénieur
SYSDYS - Systèmes dynamiques stochastiques

En TD

- continuer évaluateur, interpréteur, vérificateur de types, synthétiseur de types.
- mettre le GC dans PCF

La prochaine fois

- Composition
- Oraux des projets