ISN Informatique et société numérique Niv I

Paradigmes de programmation

Une brève introduction

Christophe Rey
christophe.rey@univ-bpclermont.fr
LIMOS – IUT d'Allier
Université Blaise Pascal

Planning et objectifs

- Jeudi 17/11/2011 : séance « théorie »
  - Introduction
  - Concepts de programmation
  - Notion de sémantique
    - Définition d'un langage noyau simple
    - Exemples d'utilisation
  - Vers une classification des paradigmes
    - Notion de programmes déclaratifs
    - Quelques catégories
- Jeudi 8/12/2011 : séance « application »

Planning et objectifs

- Jeudi 17/11/2011 : séance « théorie »
- Jeudi 8/12/2011 : séance « application »
  - Quelques mots sur l'efficacité (complexité) calculatoire (si on a le temps)
  - Etude de cas
    Algorithmique → prog structurée → POO en python
    - Présentation des principes généraux
    - Réflexions sur l'enseignement de ce paradigme très utilisé
    - Que reste-t-il alors des notions vues au premier cours ?

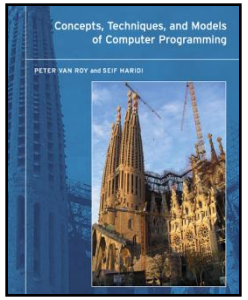
[Introduction]

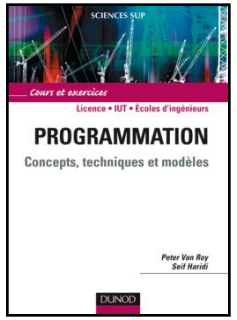
pptPlex Section Divider

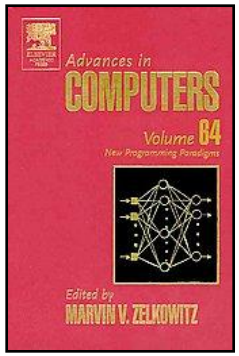
The slides after this divider will be grouped into a section and given the label you type above. Feel free to move this slide to any position in the deck.

Sources

- **Concepts, Techniques, and Models of Computer Programming**
  Peter Van Roy and Seif Haridi
  Version « brouillon » téléchargeable sur le site citeseerx
- **Programming Paradigms for Dummies**
  Roy, Peter van (2009). **Programming Paradigms for Dummies: What Every Programmer Should Know**. In G. Assayag and A. Gerzso (eds.) *New Computational Paradigms for Computer Music*, IRCAM/Delatour, France.
  http://www.info.ucl.ac.be/~pvr/VanRoyChapter.pdf
- **Les principaux paradigmes de programmation.** Peter Van Roy. Présentation à l'UPMC janvier 2008
- **Les cours** de Peter Van Roy. 2007.
  http://www.info.ucl.ac.be/~pvr/coursfrancais.html







Avertissement

- Les transparents de cette présentation représentent un résumé, à ma façon, de certaines parties des références précédentes.
- Certains transparents sont repris tels quels des cours cités précédemment.
- A l'exception des algorigrammes, le contenu est de Peter Van Roy et Seif Haridi.

Programmer ?

- Etendre ou modifier les fonctionnalités d'un système
- Faites par
  - Non spécialistes
  - Spécialistes
- Construction de systèmes logiciels Processus itératif
  - Analyse
  - Conception
  - Réalisation
  - Tests

Programmation

Science et technologie

- La programmation comme une technologie
  - Des outils
  - Des techniques pratiques
  - Des standards
- La programmation comme une science
  - Une théorie (vaste, profonde)
  - Possibilité d'étudier scientifiquement les programmes
- Sans la technologie, des mathématiques
  ➔ connaître les outils, c'est préparer le présent
- Sans la science, de l'artisanat
  ➔ connaître les concepts, c'est préparer l'avenir

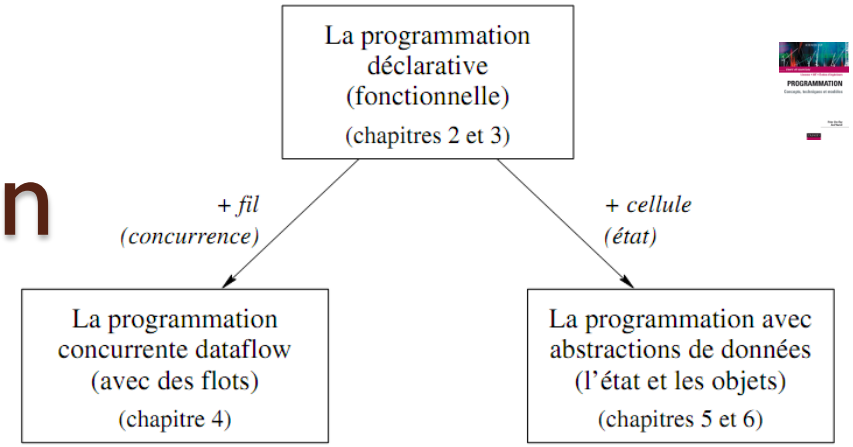
Notions principales

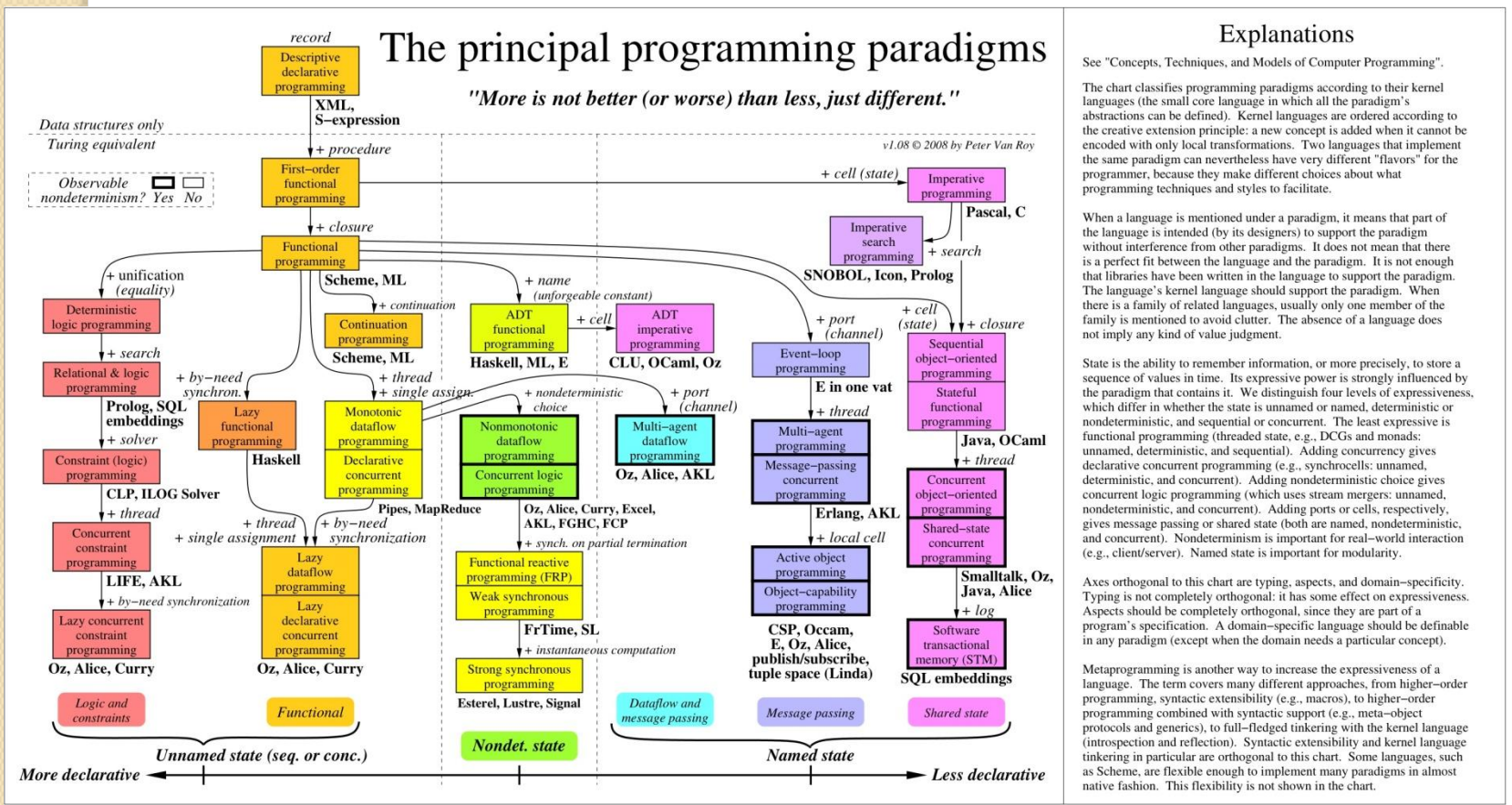
- Paradigmes de programmation
  - Constitués par un ensemble de concepts de programmation
  - Enrichis d'une sémantique
    → modèle de programmation
    → permet de raisonner sur les programmes
    - Exactitude
    - Complexité calculatoire
- Méthodologie de développement

Principaux concepts de programmation

- Les variables
  - Définition
  - Identificateur
  - Case mémoire
- Les fonctions
  - Définition
  - Récursion
  - L'abstraction fonctionnelle
- La fermeture
- Les enregistrements
- Les listes
  - Définition
  - Exemples
- Les capacités
  - Encapsulation
  - Sécurité
- Exécution paresseuse
- Canaux de communication
- La programmation d'ordre supérieure
- La concurrence
  - Définition
  - Le dataflow
- L'état explicite
  - Définition
  - Cellule de mémoire
- Les objets et classes
  - Définition
- Le non-déterminisme et le temps
- Envoi de messages
- Objets actifs
- Etats partagés et verrous
- Moniteurs et transactions
- …

Paradigmes de programmation





Explanations

See "Concepts, Techniques, and Models of Computer Programming".

The chart classifies programming paradigms according to their kernel languages (the small core language in which all the paradigm's abstractions can be defined). Kernel languages are ordered according to the creative extension principle: a new concept is added when it cannot be encoded with only local transformations. Two languages that implement the same paradigm can nevertheless have very different "flavors" for the programmer, because they make different choices about what programming techniques and styles to facilitate.

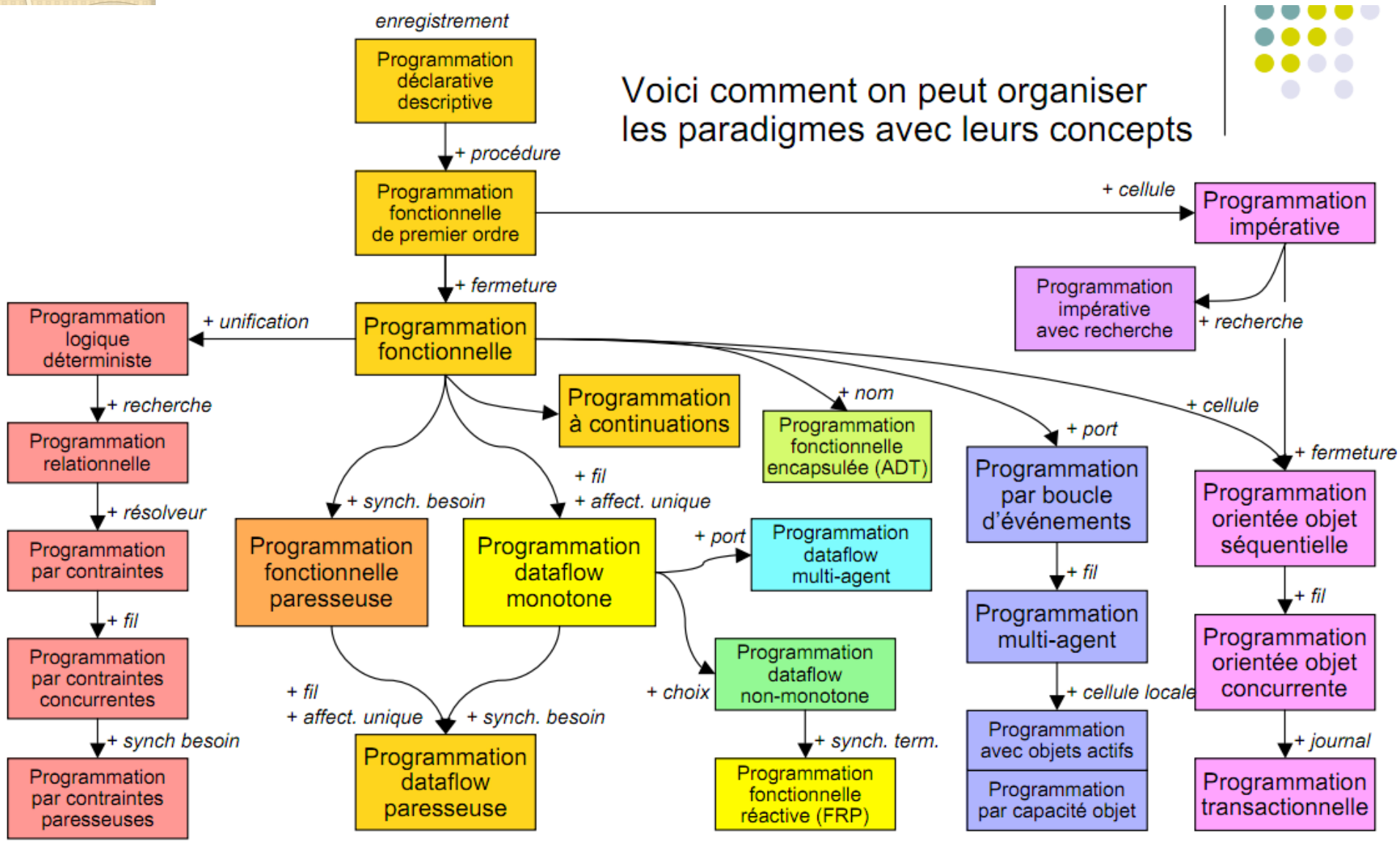
When a language is mentioned under a paradigm, it means that part of the language is intended (by its designers) to support the paradigm without interference from other paradigms. It does not mean that there is a perfect fit between the language and the paradigm. It is not enough that libraries have been written in the language to support the paradigm. The language's kernel language should support the paradigm. When there is a family of related languages, usually only one member of the family is mentioned to avoid clutter. The absence of a language does not imply any kind of value judgment.

State is the ability to remember information, or more precisely, to store a sequence of values in time. Its expressive power is strongly influenced by the paradigm that contains it. We distinguish four levels of expressiveness, which differ in whether the state is unnamed or named, deterministic or nondeterministic, and sequential or concurrent. The least expressive is functional programming (threaded state, e.g., DCGs and monads: unnamed, deterministic, and sequential). Adding concurrency gives declarative concurrent programming (e.g., synchrocells: unnamed, deterministic, and concurrent). Adding nondeterministic choice gives concurrent logic programming (which uses stream mergers: unnamed, nondeterministic, and concurrent). Adding ports or cells, respectively, gives message passing or shared state (both are named, nondeterministic, and concurrent). Nondeterminism is important for real−world interaction (e.g., client/server). Named state is important for modularity.

Axes orthogonal to this chart are typing, aspects, and domain−specificity. Typing is not completely orthogonal: it has some effect on expressiveness. Aspects should be completely orthogonal, since they are part of a program's specification. A domain−specific language should be definable in any paradigm (except when the domain needs a particular concept).

Metaprogramming is another way to increase the expressiveness of a language. The term covers many different approaches, from higher−order programming, syntactic extensibility (e.g., macros), to higher−order programming combined with syntactic support (e.g., meta−object protocols and generics), to full−fledged tinkering with the kernel language (introspection and reflection). Syntactic extensibility and kernel language tinkering in particular are orthogonal to this chart. Some languages, such as Scheme, are flexible enough to implement many paradigms in almost native fashion. This flexibility is not shown in the chart.

Paradigmes de programmation



Raisonnement sur les programmes

- Savoir si un programmer est correct (exact) ou non
  - Tests avec valeurs connues
  - Raisonnement sur le programme
    - Sémantique du langage : modèle mathématique définissant exactement ce que font les opérations du programme
    - Spécification du programme : définition mathématique des entrées du programme et des résultats calculés
    - Démonstration : techniques mathématiques pour montrer que le programme satisfait la spécification (e.g. par récurrence)
    - Hypothèse : le système sur lequel s'exécute le programme est bien implémenté
- Evaluer la complexité calculatoire d'un programme
  - Complexité temporelle : avoir une idée du temps d'exécution du programme en fonction de la taille de l'entrée
  - Complexité spatiale : avoir une idée de l'espace mémoire occupé par le programme
  - En général : complexité temporelle au pire (celle du pire cas)
  - Calculs approximatifs souvent !!

Pour raisonner

- Modèle de calcul
  - Une machine abstraite
  - Un système formel
    - Syntaxe : définition d'un langage
    - Sémantique : comment les phrases de ce langage (expressions et instructions) sont exécutées par la machine abstraite.
- Sémantique
  - Opérationnelle utilisée dans un « modèle de calcul »
  - Autres
    - Logique, ensembliste
    - Axiomatique
    - Dénotationnelle
- Modèle de programmation
  - Ensemble de techniques de programmation et de principes de conception pour écrire des programmes dans le langage du modèle de calcul.
- On peut alors raisonner sur les programmes
  - augmenter leur fiabilité (➔ exactitude)
  - calculer leur efficacité (➔ complexité)

Méthodologies de développement

1. Comprendre ce que fait le programme
   ➔ texte, exemple, dessin, …
2. Algorithme intuitif
   a) Faire ressortir les structures algorithmiques (tests, boucles, appels récursifs)
   b) Syntaxe informelle
      ➔ organigrammes
3. Développement descendant
   a) Programmer la fonction principale en supposant qu'elle existe
   b) Puis programmer les fonctions auxiliaires
4. Tests ascendants
   a) D'abord les fonctions auxiliaires
   b) Puis la fonction principale
5. Développement itératif:
   a) Retour à l'étape 1.
   b) Notion de feedback