
Sémantique des langages de programmation

Master 1 - LP - MPRI

Delia Kesner
PPS, Université Paris-Diderot

Email : kesner@pps.univ-paris-diderot.fr
URL : www.pps.univ-paris-diderot.fr/~kesner

Motivations

Programming
Languages
Semantics

Informations générales

- Partiel Obligatoire : mars
- Note module : (Partiel + Examen)/ 2
- 2eme session : Examen

1

Syntax vs Semantics

Syntax :

$$x \neq 0$$

Semantics :

"Every positive natural number is different from zero"

2

3

Different techniques :

Denotational semantics

4

Result of this function : the state of the memory **after** execution

P1:a=1; b=0; P2:a=1; P3:b=0; a=1; P4:a=1; b=1;
 b=0;

6

program = transition system

P1: a=1; b=0; P2: a=1; P3: b=0; a=1;
 b=0;

☹ non equivalent programs may have the same result.

5

$$\{P\} \text{Prog} \{Q\}$$
$$\{vrai\} \quad a = 1; b = 0; \quad \{a \quad 0 \quad b \quad 0\}$$

$\{vrai\} \quad b = 0; a = 1; \quad \{a = 0 \quad b = 0\}$

$$\{vrai\} \quad a = 1; b = 1; \quad \{a \quad 0 \quad b \quad 0\}$$

7

Relation between different semantics

Two syntactic equivalent programs are operational equivalent.

Two operational equivalent programs are denotational equivalent.

Two denotational equivalent programs are axiomatic equivalent.

The converse implications are in general false.

8

Example I : String rewriting

Rewriting system :

<i>vert</i>	<i>orange</i>
<i>orange</i>	<i>rouge</i>
<i>rouge</i>	<i>vert</i>

Reduction sequence :

vert orange rouge vert orange ...

10

Operational Semantics

program = transition system

Rewriting systems are a natural tool to model transitions between objects.

9

Example II : String rewriting

Rewriting system :

•	•
• •	• •
•	•

Reduction Sequence :

• •	• •	• •
• • •	• • •	• • •

11

Example III : Term rewriting

Rewriting system for Peano arithmetic :

$$\begin{array}{ll} 0 + y & y \\ s(x) + y & s(x + y) \\ 0 \cdot y & 0 \\ s(x) \cdot y & (x \cdot y) + y \end{array}$$

Reduction sequence :

$$\begin{aligned} \text{"2" "3"} &= \underline{s(s(0)) \quad s(s(s(0)))} \\ &\underline{s(0) \quad s(s(s(0)))} + s(s(s(0))) \\ &\underline{0 \quad s(s(s(0)))} + s(s(s(0))) + s(s(s(0))) \\ &\underline{0 + s(s(s(0)))} + s(s(s(0))) \\ &\underline{s(s(s(0)))} + s(s(s(0))) \\ &\underline{s(s(s(0)) + s(s(s(0))))} \\ &\underline{s(s(s(0) + s(s(s(0))))} \\ &\underline{s(s(s(0 + s(s(s(0))))} \\ &s(s(s(s(s(s(0)))))) = \text{"6"} \end{aligned}$$

12

13

Example IV : Equational Programming

Rewriting system :

$$\begin{array}{ll} nil[a/y] & nil \\ cons(a, x)[a/y] & cons(y, x[a/y]) \\ cons(b, x)[a/y] & cons(b, x[a/y]) \end{array}$$

Reduction sequence :

$$\begin{aligned} &cons(a, cons(b, cons(a, cons(b, nil))))[a/c][b/d] \\ &cons(c, cons(d, cons(c, cons(d, nil)))) \end{aligned}$$

14

Example V : Logical reasoning

Rewriting system :

$$\begin{array}{ll} p \quad q & \neg p \quad q \\ \neg(p \quad q) & \neg p \quad \neg q \\ \neg(p \quad q) & \neg p \quad \neg q \\ \neg \neg p & p \end{array}$$

Reduction sequence :

15

$$\begin{array}{c}
\neg(\neg(\neg p \quad q) \quad r) \\
\neg(\neg(\neg \neg p \quad q) \quad r) \\
\neg(\neg(p \quad q) \quad r) \\
\neg((\neg p \quad \neg q) \quad r) \\
\neg(\neg(\neg p \quad \neg q) \quad r) \\
\neg((\neg \neg p \quad \neg \neg q) \quad r) \\
\neg((p \quad \neg \neg q) \quad r) \\
\neg((p \quad q) \quad r) \\
\neg(p \quad q) \quad \neg r \\
(\neg p \quad \neg q) \quad \neg r
\end{array}$$

16

Example VI : Functional Programming

The λ -calculus [Church]

$$\begin{array}{lll}
(\lambda x.M)N & \rightarrow & M\{x/N\} \\
(\lambda x.M) x & \rightarrow & M \\
(SP) & \lambda_1(M), \lambda_2(M) & \rightarrow M
\end{array}$$

17

Example VII : Object-Oriented Programming

The λ -calculus [Abadi & Cardelli]

An **objet** O is a collection of **methods** $l_i \rightarrow self_i.B_i \quad i = 1..n$.

Method invocation is given by the rewriting rules :

$$O.l_j \rightarrow B_j\{self_j/O\}$$

18

Example VIII : Mathematical reasoning

$$T \vdash \lambda x.X \quad T \vdash \lambda x.X$$

$$T \vdash \{ \lambda x.A \mid P \} \quad T \vdash A \mid P \{ \lambda x.T \}$$

19

Example IX : Pattern Matching

$$\begin{array}{ll} \text{case}(c_1(L), x_1.M_1, \dots, x_n.M_n) & M_1\{x_1/L\} \\ \vdots & \vdots \\ \text{case}(c_n(L), x_1.M_1, \dots, x_n.M_n) & M_n\{x_n/L\} \end{array}$$

20

Example X : OCAML Programs

```
let rec length l = match l with
  [] -> 0
  | h::t -> length t + 1 ;;
```

```
let rec append l1 l2 = match l1 with
  [] -> l2
  | h::t -> h::append t l2;;
```

```
let rec map l f = match l with
  [] -> []
  | h::t -> (f h):: map t f ;;
```

21

Example XI : Concurrent Programming

The λ -calculus [Milner & Parrow & Walker]

$$- t.P \mid x.Q \qquad P \mid Q\{x/t\}$$

22

Example XII :XSLT

A language to transform (by rewriting) XML documents.

See <http://www.w3.org/Style/XSL/>

23

Example XIV : development of plant structures



See <http://algorithmicbotany.org/> for more details.

24

Main components of the rewriting model

- Objects
- Substitution
- Matching

26

Formal definition of rewriting

Given a set of **objects** A , (**abstract**) rewriting is a **relation** $A \times A$.

String rewriting : A is a set of words.

First-order rewriting : A is a set of algebraic terms.

Higher-order rewriting : A is a set of higher-order terms.

Graph rewriting : A is a set of graphs.

25

Typical questions concerning a rewriting model

Consider a rewriting sequence

$$t_1 \quad R \quad t_2 \quad R \quad t_3 \quad R \quad \dots$$

- Is this computation **terminating**? (always? sometimes?)
- Is there a **result** (e.g. canonical form)?
- Is there **uniqueness** of results?

27

(Part of the) History

- (Thue-1914) String rewrite systems
- (Church-1936) Lambda calculus
- (Gorn-1967) Term rewrite systems
- (Klop-1980) Combinatory reduction systems

28

- Lambda termes, substitutions, alpha-conversion
- Réduction beta et égalité beta
- Confluence du lambda calcul (réductions parallèles)
- Lambda calcul typé : type, jugement de type, dérivation de typage
- Propriétés : Unicité, décidabilité et préservation du typage, normalisation forte du lambda calcul simplement typé (méthode de réductibilité et preuve arithmétique)
- **Sémantique opérationnelle**
 - Sémantique opérationnelle structurée (petits pas)
 - Sémantique opérationnelle naturelle (grands pas)
 - Exemple évaluation des nombres binaires
 - Sémantique du lambda calcul : appel par nom et par valeur, à petits pas et grands pas

30

Plan du cours

- **Introduction**
- **Notions mathématiques**
 - Relations bien fondées : définitions et exemples
 - Principe d'induction bien fondée
 - Composition de relations bien fondées : lexicographique, produit, multi-ensemble, etc
 - Notions de réécriture abstraite
 - Notions de Church-Rosser, confluence, confluence faible, confluence forte, normalisation faible et normalisation forte
 - Quelques théorèmes fondamentaux
- **Calculs fonctionnels**
 - Introduction au lambda calcul

29

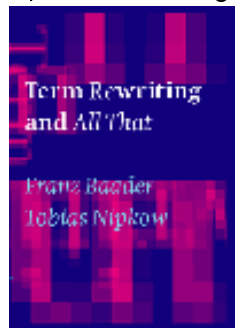
- Propriétés : déterminisme, correspondance entre petits pas et grands pas, héritage de la préservation de types et de la normalisation forte
- **Calculs algébriques**
 - Termes et Sigma algèbres
 - Homomorphismes et substitutions
 - Théorème de Birkhoff
 - Réécriture
 - Égalité engendrée par une relation de réécriture
 - Quelques techniques pour montrer la confluence
 - Quelques techniques pour montrer la terminaison
- **Machines à environnement pour l'appel par nom**
 - Évaluateur simple
 - Machine de Krivine

31

- **Sémantique dénotationnelle**
 - L'approche dénotationnelle de la sémantique
 - Sémantique d'un langage fonctionnel simple
 - Modéliser la non terminaison : les ordres partiels complets (CPO)

32

- Pour les calculs algébriques :
Term Rewriting and All That. 1998.
 Franz Baader, Tobias Nipkow. Cambridge University Press



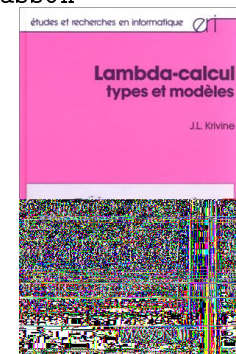
34

Bibliographie

- *Transparents et tableau*
 (consulter www.pps.jussieu.fr/~kesner régulièrement)

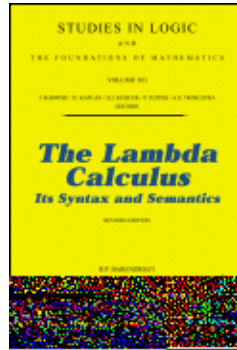
33

- Pour les calculs fonctionnels :
Lambda-calcul, types et modèles. 1997.
 Jean-Louis Krivine. Masson



35

The lambda calculus : its Syntax and Semantics. 1984. Henk Barendregt. North-Holland



Notions mathématiques pour la sémantique

36

Ensembles

Notions mathématiques de base

Relations

Définition : Une **relation n-aire** sur $A_1 \dots A_n$ est un sous-ensemble de $A_1 \times \dots \times A_n$.

Définition : Soit $R \subseteq A \times A$ une relation **binaire**.

- R est **réflexive** ssi pour tout $x \in A$, $(x, x) \in R$.
 R est **irréflexive** ssi pour tout $x \in A$, $(x, x) \notin R$.
- R est **symétrique** si pour tout $x, y \in A$, $(x, y) \in R$ implique $(y, x) \in R$.
 R est **anti-symétrique** si pour tout $x, y \in A$, $(x, y) \in R$ et $(y, x) \in R$ implique $x = y$.
- R est **transitive** si pour tout $x, y, z \in A$, $(x, y) \in R$ et $(y, z) \in R$ implique $(x, z) \in R$.

40

Exemples

Exemple : La relation $=$ sur les entiers naturels est réflexive, la relation $>$ sur les entiers naturels est irréflexive.

Exemple : La relation $=$ sur les ensembles est symétrique, la relation \subseteq sur les entiers naturels est anti-symétrique.

Exemple : La relation \subseteq sur les ensembles est transitive.

42

Notations

- $(x, y) \in R$ peut s'écrire aussi $x R y$.
- On peut utiliser un symbole à la place de R :
Ainsi par exemple, si \sim est une relation, alors $(x, y) \in \sim$ s'écrit $x \sim y$.
- On écrit $y \sim x$ lorsque $x \sim y$.

41

Équivalence et Congruence

Définition :

- R est une **équivalence** si elle est réflexive, symétrique et transitive.

Exercice : Montrer que $\sim = \{(x, y) \mid 3 \text{ est diviseur de } x - y\}$ est une équivalence.

- R est une **congruence** p.r. à f si R est une équivalence compatible avec f , c'est à dire, si $a_1 R b_1 \dots a_n R b_n$ implique $f(a_1, \dots, a_n) R f(b_1, \dots, b_n)$.

Exercice : Montrer que $\sim = \{(x, y) \mid 3 \text{ est diviseur de } x - y\}$ est une congruence par rapport à $+$ et à \cdot .

43

Classes d'équivalence

La **classe d'équivalence** de $a \in A$ par rapport à une équivalence R est l'ensemble $[a]_R = \{b \in A \mid aRb\}$.

44

Exemple : Soit $A = \{Paris, Lyon, Toulouse\}$ et

$R = \{(Paris, Lyon), (Paris, Toulouse), (Lyon, Paris), (Toulouse, Paris)\}$,

$R^2 = \{(Paris, Paris), (Lyon, Lyon), (Toulouse, Toulouse), (Lyon, Toulouse), (Toulouse, Lyon)\}$,

Calculer R^3 .

46

Composition de relations

Définition : Si $R \subseteq A \times B$ et $S \subseteq B \times C$, alors la **composition** de S avec R est une relation dans $A \times C$ t.q.

$S \circ R = \{(x, y) \in A \times C \mid \exists z \in B (x, z) \in R \text{ et } (z, y) \in S\}$.

Définition : Soit $R \subseteq A \times A$. On note R^n la **n-composition** de R avec elle-même par induction comme suit :

$$\begin{aligned} R^0 &= \{(a, a) \mid a \in A\} \\ R^{n+1} &= R^n \circ R = R \circ R^n = \underbrace{R \circ \dots \circ R}_{n+1 \text{ fois}} \end{aligned}$$

45

Les clôtures

Définition : La **clôture transitive** d'une relation R est donnée par

$$R^+ = \bigcup_{n=1} R^n$$

La **clôture réflexive et transitive** d'une relation R est donnée par

$$R = \bigcup_{n=0} R^n = R^+ \cup R^0$$

Exemple : Dans l'exemple d'avant, $R = A \times A$.

47

Fonctions

Définition : Une fonction f entre deux ensembles A et B , notée $f: A \rightarrow B$, est une relation sur $A \times B$ t.q. pour tout x, y, z si $(x, y) \in f$ et $(x, z) \in f$, alors $y = z$.

Notation : On écrit $f(x)$ pour dénoter l'unique élément y t.q. $(x, y) \in f$ et $f(C) = \{y \in B \mid \exists x \in C, f(x) = y\}$.

On note id_A la fonction identité sur A donnée par $id_A(x) = x$.

Définition : Soit $f: A \rightarrow B$ une fonction.

Image : $f(A) = \{y \in B \mid \exists x \in A, (x, y) \in f\}$

– L'image de f est $Im(f) = \{y \in B \mid \exists x \in A, (x, y) \in f\}$

– L'inverse de f est $f^{-1} = \{(y, x) \in B \times A \mid (x, y) \in f\}$

48

Par induction, voir
la Section suivante

Exercice : Soit $n > 0$. Montrer que $f^n = f^{n-1} \circ f$.

50

Composition de fonctions

Définition :

– La composition de $f: B \rightarrow C$ avec $g: A \rightarrow B$ est la fonction $f \circ g: A \rightarrow C$, où $f \circ g(x) = f(g(x))$.

Exemple : $f(x) = x^2$, $g(x) = x + 4$, $f \circ g(x) = (x + 4)^2$,
 $g \circ f(x) = x^2 + 4$.

– La n -composition de f avec elle-même, notée f^n , est défini par récurrence sur n :

– Si $n = 0$, alors $f^0 = id$

– Si $n > 0$, alors $f^n = f \circ f^{n-1}$

Exemple : $f(x) = x + 2$, $f^0(x) = x$, $f^1(x) = x + 2$,
 $f^2(x) = x + 4$, $f^3(x) = x + 6$, ..., $f^n(x) = x + 2.n$.

49

Propriétés des fonctions

Définition : Une fonction $f: A \rightarrow B$ est **injective** ssi pour tout $x, y \in A$, $f(x) = f(y)$ implique $x = y$.

Exemple : $f(x) = x + 2$ sur les entiers est injective.

$f(x) = x \setminus \{3\}$ sur les ensembles d'entiers n'est pas injective. Ainsi $f(\{2, 3, 4\}) = f(\{2, 4\})$ mais $\{2, 3, 4\} \neq \{2, 4\}$.

Définition : Une fonction $f: A \rightarrow B$ est **surjective** ssi pour tout $y \in B$ il existe $x \in A$ tel que $f(x) = y$.

Exemple : $f(x) = x \text{ div } 2$ sur les entiers naturels est surjective.
 $f(x) = x + 2$ sur les entiers naturels n'est pas surjective.

51

Définition : Une fonction est **bijective** ssi elle est injective et surjective.

Exemple : Soit A l'ensemble de mots de longueur 3 contenant uniquement 0 et 1. Soit $B = \{0 \dots 7\}$. Soit $f("b_2b_1b_0") = b_2 \cdot 2^2 + b_1 \cdot 2^1 + b_0 \cdot 2_0$. Cette fonction est injective et surjective, donc bijective.

52

Préordres, ordres

Définition :

- Un **préordre** est une relation réflexive et transitive.

Exemple :

$$R = \{(2, 2), (3, 3), (4, 4), (3, 2), (2, 3), (2, 4), (3, 4)\}.$$

- Un **ordre** ou **ordre partiel** est une relation réflexive, anti-symétrique et transitive.

Notation :

Exemple : R n'est pas un ordre car $(3, 2), (2, 3)$ mais $2 \neq 3$. $S = \{(2, 2), (3, 3), (4, 4), (2, 3), (2, 4), (3, 4)\}$ est un ordre.

Définition : Un **ordre strict** est une relation irreflexive et transitive.

54

Fonction caractéristique

Définition : Soit A un ensemble inclus dans un univers U . La **fonction caractéristique** de A dans U est la fonction

$\chi_A : U \rightarrow \{0, 1\}$ telle que

$$\chi_A(a) = 1 \text{ ssi } a \in A$$

53

Notation : $>$

Exemple : $>$ sur les entiers, \subseteq sur les ensembles.

Définition : Un ordre strict est **bien fondé** ssi il n'existe aucune chaîne infinie (i.e., de la forme $a_0 > a_1 > a_2 > \dots$).

Exemple : $>$ sur les entiers naturels est bien fondé. $>$ sur tous les entiers n'est pas bien fondé. \subseteq sur les ensembles est bien fondé.

55

Majorants/minorants et bornes supérieures/inférieures

Soit E un ensemble muni d'un ordre \leq . Soit $A \subseteq E$.

Définition :

Un **majorant** de A est un $x \in E$ t.q. pour tout $y \in A$, $y \leq x$.

Un **minorant** de A est un $x \in E$ t.q. pour tout $y \in A$, $x \leq y$.

La **borne supérieure** de A , notée $\sup(A)$, est le plus petit des majorants de A (si z est un majorant de A alors $\sup(A) \leq z$).

La **borne inférieure** de A , notée $\inf(A)$, est le plus grand des minorants de A (si z est un minorant de A alors $z \leq \inf(A)$).

Exemple : Soit $A = \{1, \dots, 10\}$. Tous les entiers dans $\{10, \dots\}$ sont des majorants de A et 10 est la borne supérieure.

56

Exemple : Si $a \leq c$, $a \leq d$, $b \leq c$, $b \leq d$, alors a et b sont des minorants, mais comme ils sont incomparables il n'y a pas de borne inférieure.

Exemple : Si E_i sont des ensembles dans $P(E)$, alors le sup est $\bigcup_i E_i$ et le inf $\bigcap_i E_i$.

57

Fonctions monotones et points fixes

Définition : Soit $f : A \rightarrow B$ une fonction et soient \leq_A, \leq_B deux ordres sur A et B respectivement.

La fonction f est **monotone** ssi $x \leq_A y$ implique $f(x) \leq_B f(y)$.

Exemple : $f(x) = x + 3$.

Définition : Soit $f : A \rightarrow A$ une fonction.

Un **point fixe** de f est un élément $x \in A$ t.q. $f(x) = x$.

Exemple : Soit $f(x) = x^2$. Alors $x = 1$ est un point fixe.

Le **plus petit point fixe** de f est $\inf(\{x \in A \mid f(x) = x\})$.

Le **plus grand point fixe** de f est $\sup(\{x \in A \mid f(x) = x\})$.

58

Définitions Inductives

Définitions inductives en informatique

- Syntaxe concrete
- Syntaxe abstraite
- Règles de typage
- Règles d'évaluation

60

Notation

Les règles d'inférence sont notées

$$\frac{\text{Hypothèse}_1 \dots \text{Hypothèse}_n}{\text{Conclusion}} \text{ (Nom de la règle)}$$

- Conclusion est une assertion
- Hypothèse₁ ... Hypothèse_n sont des assertions
- En général $n \geq 0$. Si $n = 0$ la règle est un **axiome**

62

Le principe

Une définition inductive est caractérisée par :

- Une ou plusieurs **assertions**
- Un ensemble de **règles** d'inférence pour dériver ces assertions

Exemple :

- Assertion : "**X est naturel**" ou "**X nat**"
- Règles d'inférence :

R1 : **0 est naturel**

R2 : Si **n est naturel**, alors **succ(n) est naturel**.

61

Exemple (règle unaire)

Les entiers naturels

$$\frac{}{0 \text{ est naturel}} (\text{Nat}0) \quad \frac{n \text{ est naturel}}{\text{succ}(n) \text{ est naturel}} (\text{Nat}+)$$

63

Exemple (règle binaire)

Les arbres binaires

$$\frac{}{vide \text{ est un arbre binaire}} \text{ (Abin-nil)}$$

$$\frac{A_1 \text{ est un arbre binaire} \quad A_2 \text{ est un arbre binaire}}{node(A_1, A_2) \text{ est un arbre binaire}} \text{ (Abin-)}$$

64

Exemple

Les mots sur un alphabet A

$$\frac{}{\text{mot}} \quad \frac{a \quad A \quad n \text{ mot}}{a.n \text{ mot}}$$

65

Exemple (plusieurs axiomes, règles unaires et binaires)

Les expressions de la logique propositionnelle sur l'alphabet A

$$\frac{p \quad A}{p \text{ expr}}$$

$$\frac{A_1 \text{ expr} \quad A_2 \text{ expr}}{A_1 \quad A_2 \text{ expr}} \quad \frac{A_1 \text{ expr} \quad A_2 \text{ expr}}{A_1 \quad A_2 \text{ expr}}$$

$$\frac{A_1 \text{ expr} \quad A_2 \text{ expr}}{A_1 \quad A_2 \text{ expr}} \quad \frac{A \text{ expr}}{\neg A \text{ expr}}$$

66

Exemple (plusieurs assertions)

Les forêts de type T

$$\frac{}{a \text{ vide} \quad \text{arbre T}} \quad \frac{}{f \text{ vide} \quad \text{foret T}}$$

$$\frac{t \quad T \quad f \quad \text{foret T}}{node(t, f) \quad \text{arbre T}} \quad \frac{A \quad \text{arbre T} \quad f \quad \text{foret T}}{add(A, f) \quad \text{foret T}}$$

67

Dérivation d'une assertion

Une assertion A est **dérivable** ssi

– A est un axiome

$$\frac{}{A}$$

– ou il y a une règle de la forme

$$\frac{A_1 \quad A_n}{A}$$

telle que A_1, \dots, A_n sont dérivables

68

Exercice :

1. Montrer que $\text{succ}(\text{succ}(\text{succ}(0)))$ nat est dérivable.
2. Donner le terme qui dénote la forêt suivante et montrer comment la construire avec les règles précédentes :

avide 3 6
 / \ / | \

Ensemble inductif

Un ensemble inductif est le plus petit ensemble engendré par un système de règles d'inférence.

Preuves par Induction

70

Preuves par induction

- Induction sur les entiers
 - Induction mathématique
 - Induction complète
 - Équivalence
- Induction bien fondée
- Induction structurelle
- Induction sur un ensemble inductif

72

Exemples

$$1) \sum_{i=1}^n i = \frac{n(n+1)}{2} \quad 2) \quad n^2 = \sum_{i=1}^n (2i-1)$$

Mais comment prouver

1. “Tout entier est décomposable en produit de nombres premiers”
2. “Si n est divisible par 3, alors $\text{fib}(n)$ est pair, sinon $\text{fib}(n)$ est impair”.

74

Induction sur les entiers I (induction mathématique)

Theorem : Soit P une propriété sur les entiers. Supposons

(CB) $P(0)$,

(CI) $\forall n \in \mathbb{N}. P(n) \Rightarrow P(n+1)$,

alors $\forall n \in \mathbb{N}. P(n)$

73

Induction sur les entiers II (induction complète)

Theorem : Soit P une propriété sur les entiers. Supposons

(CB) $P(0)$,

(CI) $(\forall k \in \mathbb{N}. (k < n \Rightarrow P(k))) \Rightarrow P(n)$,

alors $\forall n \in \mathbb{N}. P(n)$

75

Équivalence des deux principes

Malgré l'apparente supériorité du deuxième principe, on prouve

Theorem : Induction mathématique et complète sont équivalentes.

76

$n < n + 1$, donc on peut appliquer l'hypothèse d'induction et en déduire qu'ils sont tous d'accord avec le professeur (qui est dans les deux), ce qui nous permet de conclure. ■

Corollary : Le professeur a toujours raison.

vrai ou faux ?

Théorème fondamental du cours

Theorem : Tous le monde est d'accord avec le professeur.

Proof. On montre, par induction sur le nombre de personnes dans l'amphi, que tout groupe de n personnes contenant le professeur est d'accord avec lui.

Cas de base : il y a seulement le professeur, trivial.

Cas inductif : on suppose l'énoncé vrai pour tout groupe de n personnes, et on le prouve pour tout groupe de $n + 1$.

Numérotons de 1 à $n + 1$ les personnes en question, de façon que le professeur soit le numéro n , et considérons le groupe A des premières n et le groupe B des dernières n personnes.

Les deux groupes contiennent le professeur et sont de taille

77

Principe d'induction bien fondée

Un ensemble A , un ordre strict $>$ et une propriété P sur A

Principe d'induction :

Si

(CB) Pour tout élément minimal $y \in A$, $P(y)$.

(CI) $((\forall z \in A. (z < x \Rightarrow P(z))) \Rightarrow P(x))$

"le fait que $P(z)$ soit vérifiée pour tout élément $z < x$ implique $P(x)$ "

alors

$\forall x \in A. P(x)$

78

79

Ce principe est-il toujours bien défini ?

Soit $>$ un ordre strict.

Theorem :

Si $>$ est **bien fondé**, alors le principe d'induction est correct.

Theorem :

Si le principe d'induction est correct, alors $>$ est **bien fondé**.

Corollary : Le principe d'induction est correct pour les ensembles inductifs.

Corollary : Le principe d'induction structurelle est correct.

80

Induction sur Quelques sur d'ordres bien fondés

- Ordre lexicographique
- Ordre multi-ensemble
- Combinaisons

82

Exemples

– Les mots :

$P(m)$ est la propriété :

$$\text{concat}(\text{concat}(m, v_1), v_2) = \text{concat}(m, \text{concat}(v_1, v_2))$$

– Les arbres binaires :

$P(a)$ est la propriété : $\text{feuilles}(a) = \text{noeuds_internes}(a) + 1$

81

Ordres lexicographiques

Soit $>_{A_i}$ un ordre strict sur l'ensemble A_i .

Ordre lexicographique sur le produit de 2 ensembles :

$$(X, y) >_{lex} (X, y) \text{ ssi } (X >_{A_1} X) \text{ ou } (X = X \text{ et } y >_{A_2} y)$$

Example :

$$(4, "abc") >_{lex} (3, "abc") >_{lex} (2, "abcde") >_{lex} (2, "bcde") >_{lex} (2, "e") >_{lex} (1, "e") >_{lex} (0,)$$

83

Ordre lexicographique sur le produit de n ensembles

Si chaque $>_{A_i}$ est un ordre strict sur l'ensemble A_i , alors $>_{lex}$ est un ordre strict qui permet de comparer deux n -uplets de la manière suivante :

$$(x_1, \dots, x_n) >_{lex} (x'_1, \dots, x'_n) \text{ ssi } \exists j \in \{1, \dots, n\} \text{ tel que } (x_j >_{A_j} x'_j \text{ and } \forall i < j, x_i = x'_i)$$

Theorem : Si chaque $>_{A_i}$ est un ordre strict bien fondé sur A_i , alors l'ordre lexicographique $>_{lex}$ sur le produit de $A_1 \times \dots \times A_n$ est un ordre strict bien fondé sur $A_1 \times \dots \times A_n$.

Avertissement : $>_{lex}$ n'est pas l'ordre du dictionnaire !!

84

Les multi-ensembles

Définition : Soit A un ensemble. Un **multi-ensemble** de base A est une fonction $M: A \rightarrow \mathbb{N}$. Le multi-ensemble M est **fini** si $M(x) > 0$ seulement pour un nombre fini d'éléments de A .

Notation : $\{a, a, b\}$.

86

Exemple : la fonction d'Ackerman

Montrer par induction que la fonction suivante termine.

$$\begin{aligned} \text{Ackerman}(0, n) &= n+1 \\ \text{Ackerman}(m+1, 0) &= \text{Ackerman}(m, 1) \\ \text{Ackerman}(m+1, n+1) &= \text{Ackerman}(m, \text{Ackerman}(m+1, n)) \end{aligned}$$

85

Ordres multi-ensembles

Définition : $M >_{mul} N$ ssi N s'obtient à partir de M en appliquant la règle suivante un nombre fini de fois : enlever un élément x de M et le remplacer par un nombre fini d'éléments plus petits que x (par rapport à l'ordre $>$).

Notation :

$$\{5, 3, 1, 1\}$$

Exemple :

$$\{5, 3, 1, 1\} >_{mul} \{4, 3, 3, 1\}$$

Theorem : Si $>_A$ est un ordre strict bien fondé sur A , alors $>_{mul}$ est un ordre strict bien fondé sur les multi-ensembles de base A .

87

Exemple

Un homme possède une somme d'argent en euros. Chaque jour il procède de la façon suivante :

- soit il jette une pièce de monnaie dans une fontaine,
- ou bien il change l'un de ses billets à la banque par un nombre arbitraire de pièces de monnaie de valeur quelconque.

Montrer que ce processus termine, c'est à dire, que dans un temps fini l'homme est ruiné.