Sémantique de Niklaus

Frédéric Boulanger

CentraleSupélec

01/2020

Rappel : Niklaus

Niklaus : langage utilisé dans le cours de traitement des langages

- instructions de lecture et d'écriture sur une console
- affectation, conditionnelle, boucle *while*
- instructions exécutées en séquence

Exemple :

```
program PGCD;
var a, b;
{
    read a;
    read b;
    while(a <> b) {
        if(a > b) {
            a := a - b;
        } else {
            b := b - a;
        }
    }
    write a;
}
```

Définition d'une sémantique

Ingrédients :

- syntaxe concrète (vue dans les cours précédents)
- syntaxe abstraite
- domaine sémantique
- correspondance entre la syntaxe abstraite et le domaine sémantique.

Domaine sémantique

Logique d'ordre supérieure

- possibilité de quantifier les fonctions et les prédicats
- cadre mathématique formel
- outillé (assistant de preuve Isabelle)

Pour des raisons pratiques, on définira la syntaxe abstraite
et la correspondance sémantique dans le même cadre
par des types de données, fonctions et prédicats.

Isabelle

Assistant de preuve fondé sur un noyau (*Pure*) :

- $\Rightarrow$ constructeur de type fonction :
  $\mathrm{nat} \Rightarrow \mathrm{nat}$ = fonction de $\mathbb{N}$ vers $\mathbb{N}$.
- $\bigwedge$ quantificateur universel : $\bigwedge x.\ P(x)$
- $\Longrightarrow$ implication : $P \Longrightarrow Q$ ou $P \Longrightarrow Q \Longrightarrow R$

Curryfication

$f : (x, y) \mapsto xy$ est une fonction de deux arguments.

On peut l'écrire comme : $f : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$
Mais également sous forme curryfiée : $f : \mathbb{N} \rightarrow (\mathbb{N} \rightarrow \mathbb{N})$
On interprète alors $f(x, y)$ comme : $f(x)(y)$
$f(x)$ rendant une fonction que l'on applique à y.

En Isabelle, on aura : $f :: nat \Rightarrow nat \Rightarrow nat$

Exemple de curryfication

Règles de déduction

$$\dfrac{\begin{matrix}[x :: \alpha] \\ \vdots \\ b\ x :: \beta\end{matrix}}{\lambda x.\ b\ x :: \alpha \Rightarrow \beta}\ (\Rightarrow I)$$

$$\dfrac{b :: \alpha \Rightarrow \beta \quad a :: \alpha}{b\ a :: \beta}\ (\Rightarrow E)$$

$$\dfrac{\begin{matrix}[x] \\ \vdots \\ B(x)\end{matrix}}{\bigwedge x.\ B(x)}\ (\bigwedge I)$$

$$\dfrac{\bigwedge x.\ B(x)}{B(a)}\ (\bigwedge E)$$

$$\dfrac{\begin{matrix}[A] \\ \vdots \\ B\end{matrix}}{A \Longrightarrow B}\ (\Longrightarrow I)$$

$$\dfrac{A \Longrightarrow B \quad A}{B}\ (\Longrightarrow E)$$

Isabelle/Pure

Égalité

$\equiv$

$t \equiv u$ est définie selon :

- α-conversion :
  $\lambda x.\ 2 \times x \equiv \lambda y.\ 2 \times y$
- β-reduction :
  $(\lambda x.\ 2 \times x)\ 4 \equiv 2 \times 4$
- η-conversion :
  $\lambda x.\ f\ x \equiv f$
- *refl* : $t \equiv t$, *subst* : $[\![s \equiv t; P(s)]\!] \Longrightarrow P(t)$,
  ext : $(\bigwedge x.\ f\ x \equiv g\ x) \Longrightarrow f \equiv g$,
  iff : $[\![P \Longrightarrow Q; Q \Longrightarrow P]\!] \Longrightarrow P \equiv Q$

Isabelle HOL

Logique d'ordre supérieur fondée sur *Pure*.

Calcul des propositions :

- type *bool*, avec constantes *True* et *False*
- opérateurs logiques $\wedge$, $\vee$, $\neg$, $\longrightarrow$, $\longleftrightarrow$
- égalité $=$, et tiers exclu : $a = \mathrm{True} \vee a = \mathrm{False}$

Logique du premier ordre :

- quantificateurs $\forall$ et $\exists$

Logique d'ordre supérieur : les quantificateurs s'appliquent aux fonctions et aux prédicats, les fonctions et prédicats peuvent prendre pour argument des fonctions et des prédicats.

Définition de types : entiers, listes, ensembles etc.

Exemple Faire le tutoriel

CentraleSupélec

Syntaxe abstraite – Concepts de Niklaus

Instructions

- déclaration de variable : `var a;`
- affectation : `a := a - b:`
- mise en séquence : `read a; read b;`
- alternative et boucle while : `if (c) {} else {}` et `while (c) {}`

Expressions

- entier
- variable
- opérations arithmétiques : $+$, $-$, $\times$, $\div$

Expressions booléennes

- booléen : `true`, `false`
- comparaison d'expressions : $=$, $\neq$, $<$, $\leqslant$, $>$, $\geqslant$

Syntaxe abstraite en Isabelle

Domaine sémantique et correspondance

Valeurs et opérations

Entiers de Niklaus → type `int` d'Isabelle/HOL
Noms de variable → type `string` d'Isabelle/HOL
Opérations arithmétiques → arithmétique d'Isabelle/HOL

État de la machine d'exécution

Pour simplifier, on ignore les instructions `read` et `write`
État de la machine = valeur des variables (string ⇒ int)

L'instruction en cours d'exécution dans un programme n'est pas modélisée explicitement. C'est l'opérateur de séquencement qui fait passer d'un instruction à la suivante.

L'instruction `Nop` correspond à un programme vide.

Domaine sémantique en Isabelle

CentraleSupélec