

Partielle Korrektheit

Verhält sich ein Algorithmus richtig für alle Eingaben, für die er terminiert, so spricht man von *partieller Korrektheit*.

Terminiert er außerdem für alle interessierenden Eingaben, so liegt *totale Korrektheit* vor.

Häufig kann man partielle Korrektheit unabhängig von der totalen Korrektheit und mit anderen Methoden zeigen.

Partielle Korrektheit rekursiver Funktionen

Seien A, B Mengen und $R \subseteq A \times B$ eine Relation. Sei A' eine Teilmenge von A . Wir möchten zu jedem $x \in A'$ ein $y \in B$ berechnen, sodass xRy gilt.

Satz von der partiellen Korrektheit: Sei

$$f = \mathbf{function}(x)\Phi(f, x)$$

In Φ befinde sich f nicht im Geltungsbereich einer weiteren Funktionsabstraktion außer $\mathbf{function}(x)$ (also z.B. nicht $\Phi(f, x) = \mathbf{function}(y)f(x)(y)$).

Um zu zeigen, dass für alle $x \in A' \subseteq A$ entweder $f(x)$ undefiniert ist oder $xRf(x)$ gilt, genügt es, folgendes nachzuweisen:

- ist $x \in A'$, so werden in $\Phi(f, x)$ nur Aufrufe $f(x')$ mit $x' \in A'$ getätigt.
- Für alle $x \in A'$ gilt $xR\Phi(f, x)$ unter der Annahme, dass $x'Rf(x')$ für alle in $\Phi(f, x)$ getätigten Aufrufe $f(x')$ von f .

Beispiel

$f = \text{function}(x, y)$
 if $x = 0$ **then** 0 **else**
 if x gerade **then** $2 \cdot f(x, y/2)$ **else** $2f(x, \lceil y/2 \rceil) + x$

Man zeige: für alle $x, y \in \mathbb{N}$ gilt: falls $f(x, y)$ definiert ist, dann ist
 $f(x, y) = xy$.

$f = \text{function}(x)$
 if $x = 0$ **then** 0 **else**
 if x gerade **then** $f(x/2)$ **else** $f(3x + 1)$

Man zeige: für alle $x \in D(f)$ ist $f(x) = 0$.

McCarthy's Funktion

$f = \text{function}(x:\text{int})\text{int}$

if $x > 100$ **then** $x - 10$ **else** $f(f(x + 11))$

$$f(91) = f(f(102)) = f(92) = f(f(103)) = f(93) = \dots = f(101) = 91$$

$$\begin{aligned} f(70) &= f(f(81)) = f^3(92) = f^4(103) = f^4(93) = f^4(104) = f^3(94) = \\ &f^4(105) = f^3(95) = f^4(106) = f^3(96) = f^4(107) = f^3(97) = \\ &f^4(108) = f^3(98) = f^4(109) = f^3(99) = f^4(110) = f^3(100) = \\ &f^4(111) = f^3(91) = \dots = 91 \end{aligned}$$

Man beweise: falls $x \in D(f)$, so gilt

$$f(x) = \text{if } x \geq 100 \text{ then } x - 10 \text{ else } 91.$$

Denotationale Semantik

Die W -Funktion weist OCAML-Ausdrücken abstrakte mathematische Werte zu.

Etwa Funktionsausdrücken mathematische Funktionen, also (unendliche) Mengen von Paaren.

Man bezeichnet diese Art der Semantikgebung als *denotationelle Semantik*.

Vorteile der denotationellen Semantik:

Implementierungsdetails werden versteckt, mathematische Beweismethoden (Gleichungsschließen, Abstiegsfunktion, Induktion, Satz von der partiellen Korrektheit) werden verfügbar gemacht.

Operationale Semantik

Manchmal ist die denotationale Semantik zu abstrakt:

- Effizienzbetrachtungen
- Seiteneffekte wie Ein-/Ausgabe
- Verständnisschwierigkeiten bei mathematisch nicht einschlägig gebildeten Personen.

Die *operationale Semantik* beschreibt die Semantik eines Ausdrucks durch Rechenregeln, die sukzessive angewandt werden. Wert eines Ausdrucks ist dann das Endergebnis der Rechnung.

Problem: Was soll das Ergebnis eines Funktionsausdrucks sein?

Antwort: Der Funktionsausdruck selber, wobei allerdings die Werte freier Variablen gemerkt werden müssen.

Beispiel

```
let x = 1;;  
let f = function y -> x + y;;  
let x = 2;;
```

Hier sollte `f` den Wert “`fun y -> y + x`, wobei `x=1`” haben.

Solch ein Paar aus einem Funktionsausdruck und einer Umgebung (die freie Variablen des Funktionsausdrucks bindet) bezeichnet man als *Closure*.

Operationale Semantik formal

Definition *Werte* und *Umgebungen* (im Sinne der operationalen Semantik) werden wie folgt definiert:

- Eine OCAML-Konstante ist ein Wert
- Eine *Umgebung* ist eine Menge von Bindungen $\langle x, w \rangle$ von Bezeichnern an Werte.
- Ist e ein OCAML-Ausdruck, x ein Bezeichner und U eine Umgebung, so ist $(\text{fun } x \rightarrow e, U)$ ein Wert.
- Ist e ein OCAML-Ausdruck, f und x Bezeichner und U eine Umgebung, so ist $(f = \text{fun } x \rightarrow e, U)$ ein Wert.

Beispiel

Habe x den Wert 7.

Nach den Deklarationen

```
let x = 7;;
```

```
let rec f = fun y -> if y=0 then x else f (y-1);;
```

```
let g = f;;
```

hat g den Wert

```
(f = fun y -> if y=0 then x else f (y-1), {<x,7>}).
```

Auswerterelation

Wir schreiben $U, e \rightarrow w$ um zu sagen, dass in der Umgebung U der Ausdruck e als Endergebnis den Wert w hat.

Diese *Auswerterelation* wird formal durch die folgenden Regeln definiert.

Auswerteregeln

- ist c eine Konstante, so gilt $U, c \rightarrow c$
- ist x ein Bezeichner und $\langle x, w \rangle \in U$, so gilt $U, x \rightarrow w$.
- für Funktionsausdrücke gilt $U, \text{fun } x \rightarrow e \rightarrow (\text{fun } x \rightarrow e, U')$, wobei U' die Einschränkung von U auf die freien Variablen von e ist.

Auswerteregeln

- ist op ein Infixoperator aber nicht $||$, $\&\&$, welcher eine Basisfunktion $\oplus$ bezeichnet so gilt folgendes: wenn $U, e_1 \rightarrow w_1$ und $U, e_2 \rightarrow w_2$, dann $U, e_1 \text{ op } w_1 \oplus w_2$.

Natürlich müssen w_1, w_2 im Definitionsbereich von $\oplus$ liegen.

Einstellige Basisfunktionen werden analog behandelt.

Fallunterscheidung

- Gilt $U, e_1 \rightarrow true$ und $U, e_2 \rightarrow w_2$, so auch $U, \text{if } e_1 \text{ then } e_2 \text{ else } e_3 \rightarrow w_2$.
- Gilt $U, e_1 \rightarrow false$ und $U, e_3 \rightarrow w_3$, so auch $U, \text{if } e_1 \text{ then } e_2 \text{ else } e_3 \rightarrow w_3$.
- Die Infixoperatoren $\&\&$, $||$ werden analog behandelt.

Applikation

- Gilt $U, e_1 \rightarrow w_1$ mit $w_1 = (\text{fun } x \rightarrow e, U')$ und $U, e_2 \rightarrow w_2$, so ist (in einer Nebenrechnung) ein Wert w mit $U' + \{ \langle x, w_2 \rangle \}, e \rightarrow w$ zu bestimmen. Es ist dann $U, e_1 e_2 \rightarrow w$.
- Gilt $U, e_1 \rightarrow w_1$ mit $w_1 = (f = \text{fun } x \rightarrow e, U')$ und $U, e_2 \rightarrow w_2$, so ist zunächst (in einer Nebenrechnung) ein Wert w mit $U' + \{ \langle x, w_2 \rangle, \langle f, w_1 \rangle \}, e \rightarrow w$ zu bestimmen. Es ist dann $U, e_1 e_2 \rightarrow w$.

Bindung

- Ist $U, e_1 \rightarrow w_1$, so ist zunächst (in einer Nebenrechnung) ein Wert w_2 mit $U + \{<x, w_1>\}, e_2 \rightarrow w_2$ zu bestimmen. Es ist dann $U, \text{let } x=e_1 \text{ in } e_2 \rightarrow e'$.
- Es gilt $U, \text{let rec } f=\text{fun } x \rightarrow e_1 \text{ in } e_2 \rightarrow w$, falls $U + \{<f, f=\text{fun } x \rightarrow e_1>\}, e_2 \rightarrow w$.

Beispiel

Wir wollen

```
let x=1+0 in let rec f = fun y->if y=0 then x else f(y-1) in  
    let x = 2 in f (x+x)
```

in der leeren Umgebung auswerten.

Es ist $\emptyset, 1 + 0 \rightarrow 1$ also muss

```
let rec f = fun y->if y=0 then x else f(y-1) in  
    let x = 2 in f (x+x)
```

in der Umgebung $U_1 = \{ \langle x, 1 \rangle \}$ auszuwerten. Das aber bedeutet, in der Umgebung

$$U_2 = \{ \langle x, 1 \rangle, \langle f, (f = \text{fun } y \rightarrow \text{if } y=0 \text{ then } x \text{ else } f(y-1), U_1) \rangle \}$$

den Ausdruck `let x=2 in f (x+x)` auszuwerten. Dies schließlich

Beispiel

bedeutet, in der Umgebung

$$U_3 = \{ \langle x, 2 \rangle, \langle f, (f = \text{fun } y \rightarrow \text{if } y = 0 \text{ then } x \text{ else } f(y-1), U_1) \rangle \}$$

den Ausdruck $f \ (x+x)$ auszuwerten.

Es gilt $U_3, x+x \rightarrow 4$ und

$U_3, f \rightarrow (f = \text{fun } y \rightarrow \text{if } y = 0 \text{ then } x \text{ else } f(y-1), U_1)$, also
müssen wir in der Umgebung

$$U_4 = \{ \langle x, 1 \rangle, \langle f, (f = \text{fun } y \rightarrow \text{if } y = 0 \text{ then } x \text{ else } f(y-1), U_1) \rangle, \langle y, 4 \rangle \}$$

den Ausdruck

$\text{if } y = 0 \text{ then } x \text{ else } f(y-1)$

auswerten. Der Ausdruck $y = 0$ hat den Wert *false*, es gilt also in U_4 den
Ausdruck $f \ 2$ auszuwerten...

Zusammenfassung Operationale Semantik

- Die operationale Semantik ordnet Umgebungen und Ausdrücken *Werte* zu. Diese Werte sind syntaktisch definiert.
- Rekursion wird durch wiederholte Auswertung, statt durch Rekursion auf der Meta-ebene definiert.
- Die operationale Semantik liegt näheran der tatsächlichen Implementierung als die denotationale Semantik.
- Denotationale Semantik ist günstiger für mathematische Beweismethoden (Gleichungsschließen, Induktion, Satz von der partiellen Korrektheit)

Zusammenhang operationale u. denotationale Semantik

Satz: Genau dann ist $W^\emptyset(e)$ definiert, wenn es einen Wert w gibt mit $\emptyset, e \rightarrow w$.

(Falls e vom Typ `int` ist, folgt daraus, dass beide Semantiken denselben Wert liefern. Wieso?)

Satz: Gilt $W^U(e_1) = W^U(e_2)$ für alle U , so sind e_1 und e_2 bezüglich der operationalen Semantik nicht voneinander zu unterscheiden, d.h., erhält man e_4 aus e_3 durch Ersetzen eines oder mehrerer Vorkommen von e_1 durch e_2 , dann gilt $\emptyset, e_3 \rightarrow w$ für ein w genau dann wenn $\emptyset, e_4 \rightarrow w'$ für ein w' . Ist der Typ von e_3 und e_4 gleich `int`, so gilt $w = w'$.

Bemerkung: die Umkehrung des Satzes gilt nicht: es gibt ununterscheidbare Ausdrücke, die doch nicht die gleiche denotationale Semantik haben.