

Kapitel 3.4 Termauswertung

Präzedenz- und Assoziationsregeln

Operator	Präzedenz	Assoziativität	Erklärung
<i>Funktionsapplikation</i>		links	
-	10	—	Präfix-Minus
**	9	rechts	Potenzierung bei float
* . / . * / mod	8	links	
+ . - . + -	7	links	
^	6	rechts	Verkettung
< > <= >= <> =	5	links	
not	4	—	Präfix-Verneinung
&&	3	links	
	2	links	
,	1	—	Paarbildung, <i>in praxi</i> meist geklammert
if let fun function	0	—	Quasi-Präfixoperatoren

Was bedeutet das?

Dass $* /$ oberhalb von $+ -$ steht entspricht der Punkt-vor-Strich Regel.

Vergleichsoperatoren unterhalb von arithmetischen Operatoren bedeutet, dass z.B. $i - j < 7$ bedeutet $(i - j) < 7$ und nicht etwa $i - (j < 7)$ was ein Typfehler wäre.

Das Funktionsapplikation über allem steht, bedeutet, dass man z.B.: $2 \times f(x)$ in OCAML als $2 * f\ x$ schreiben kann.

Dass `if let fun` ganz unten stehen, bedeutet, dass z.B.

```
if i < j then 0 else x - y + z
```

als

```
if i < j then 0 else (x - y + z)
```

und nicht etwa

```
(if i < j then 0 else x) - y + z
```

Werte und Umgebungen

Wir wollen formal beschreiben, was der Wert eines OCAML-Ausdrucks ist.

Um den Wert eines Ausdrucks anzugeben, muss man die Werte der in ihm enthaltenen freien Variablen kennen.

Eine Zuweisung von Werten an freie Variablen heißt *Umgebung*.

Die *Semantik* eines OCAML-Ausdrucks ist also eine Abbildung von Umgebungen auf Werte.

Umgebungen

Eine *Umgebung* ist eine Menge von *Bindungen* $\langle a, w \rangle$ wobei a ein Bezeichner ist und w ein Wert.

Während einer OCAML Sitzung wird eine Umgebung (die *aktuelle Umgebung*) sukzessive aufgebaut.

Zu Beginn ist die aktuelle Umgebung leer.

Bei der Eingabe `let  $x = t$ ;;` wird der Term t in der aktuellen Umgebung ausgewertet.

Ist die Auswertung undefiniert, so entsteht eine Fehlermeldung, bzw. falls Nichttermination der Grund ist, wird nichts ausgegeben.

Anderenfalls wird die Bindung $\langle x, w \rangle$ der aktuellen Umgebung hinzugefügt, wobei w der berechnete Wert von t ist.

Eine frühere Bindung der Form $\langle x, w' \rangle$ wird dabei entfernt.

Die aktuelle Umgebung ist also zu jeder Zeit eine *endliche partielle Funktion* von Bezeichnern nach Werten.

Beispiel

Welche Umgebung liegt nach folgenden Eingaben vor?

```
let x = 2;;  
let x = x + 1;;  
let y = x + 2;;  
let f = function z -> z - x;;  
let a = f + 1;;  
let a = f (f(x)+1);;
```

Auswertung von Termen

Sei U eine Umgebung und t ein Term.

Der Wert $W^U(t)$ des Terms t in der Umgebung U ist wie folgt rekursiv definiert:

- Ist c eine Konstante, so ist $W^U(c) = c$,
- Ist x ein Bezeichner, so ist $W^U(x) = w$, falls $\langle x, w \rangle \in U$.
Anderenfalls ist $W^U(x)$ undefiniert.
- Ist op ein Infixoperator aber nicht $||$, $\&\&$, welcher eine Basisfunktion $\oplus$ bezeichnet.

Sind $W^U(t_1)$ und $W^U(t_2)$ beide definiert, so ist

$$W^U(t_1 \text{ op } t_2) = W^U(t_1) \oplus W^U(t_2).$$

Ist auch nur einer der beiden undefiniert, so ist $W^U(t_1 \text{ op } t_2)$ undefiniert.

Einstellige Basisfunktionen wie `not` und `-` sind analog.

Auswertung von Funktionstermen

Notation:

$U + \{ \langle x_1, w_1 \rangle, \dots, \langle x_n, w_n \rangle \} = U' \cup \{ \langle x_1, w_1 \rangle, \dots, \langle x_n, w_n \rangle \}$,
wobei U' aus U durch Entfernen aller eventuell vorhandenen Bindungen
von $x_1, \dots, x_n$ entsteht.

- Sei t eine Funktionsanwendung der Form $t_1 \ t_2$. Es sei $W^U(t_1)$ die Funktion f und $W^U(t_2) = w$. Ist auch nur eines der beiden undefiniert, so ist $W^U(t)$ auch undefiniert. Ansonsten ist $W^U(t) = f(w)$. Dies kann trotzdem noch undefiniert sein, falls $w \notin D(f)$.
- Sei $t = \text{function}(x_1, \dots, x_n) \rightarrow t'$. Im Falle $n = 1$ auch ohne Klammern. Es ist $W^U(t)$ diejenige Funktion, die $(w_1, \dots, w_n)$ auf $W^{U + \{ \langle x_1, w_1 \rangle, \dots, \langle x_n, w_n \rangle \}}(t')$ abbildet. Beachte: $W^U(t)$ ist immer definiert, könnte aber die nirgends definierte Funktion sein.
- Die alternativen Notationen für Funktionsdefinitionen (`fun`, `let`) haben analoge Bedeutung.

Auswertung von `if` und `let`

- Sei t ein bedingter Term der Gestalt `if  $t_1$  then  $t_2$  else  $t_3$` . Ist $W^U(t_1) = \text{true}$, so ist $W^U(t) = W^U(t_2)$. Beachte: $W^U(t_3)$ kann dann undefiniert sein.

Ist $W^U(t_1) = \text{false}$, so ist $W^U(t) = W^U(t_3)$. Beachte: $W^U(t_2)$ kann dann undefiniert sein.

In allen anderen Fällen ist $W^U(t)$ undefiniert.

- Sei t von der Form `let  $(x_1, \dots, x_n) = t_1$  in  $t_2$` im Falle $n = 1$ auch ohne Klammern.

Sei $W^U(t_1)$ definiert und von der Form $W^U(t_1) = (w_1, \dots, w_n)$, also ein n -Tupel von Werten. Dann ist

$W^U(t) = W^{U+\{\langle x_1, w_1 \rangle, \dots, \langle x_n, w_n \rangle\}}(t_2)$. Ansonsten ist $W^U(t)$ undefiniert. Beachte: $W^U(t_1)$ muss auf jeden Fall definiert sein, selbst wenn eines der oder gar alle x_i nicht in t_2 vorkommen.

Tupel und Boole'sche Operatoren

- Sei $t = (t_1, \dots, t_n)$. Seien weiter $W^U(t_1) = w_1, \dots, W^U(t_n) = w_n$ allesamt definiert. Dann ist $W^U(t) = (w_1, \dots, w_n)$.
- $W^U(t_1 \mid\mid t_2) = W^U(\text{if } t_1 \text{ then true else } t_2)$
- $W^U(t_1 \&\& t_2) = W^U(\text{if } t_1 \text{ then } t_2 \text{ else false})$.

Zur Beachtung: Die Semantikdefinition im Skript [Kröger] ist teilweise nicht ganz richtig; maßgeblich sind daher die Folien^a.

Zur Beachtung: $t_1 \mid\mid t_2$ kann definiert sein, auch wenn t_2 undefiniert ist. Auf jeden Fall muss aber t_1 definiert sein.

^aKann sein, dass die Folien auch nicht ganz richtig sind, maßgeblich ist daher der gesunde Menschenverstand :-)

Beispiel

$$\begin{aligned} & W^{\{<x,1>\}}(\text{let } x = 2 \text{ in if } x=2 \text{ then } 0 \text{ else } 1) \\ = & W^{\{<x,2>\}}(\text{if } x=2 \text{ then } 0 \text{ else } 1) \\ = & W^{\{<x,2>\}}(0) \\ = & 0 \end{aligned}$$

Semantik von `let rec`

Eine rekursive `let`-Bindung

$$\text{let rec } f\ x = t$$

bindet f an die durch

$$F(w) = W^{U+\{<x,w>, <f,F>\}}(t)$$

rekursiv definierte Funktion.

Das gilt analog für Phrasen mit `let rec ... in` und für Funktionen mit mehreren Argumenten.

Beispiel

`let rec fakt n = if n = 0 then 1 else n * fakt(n-1)`

Sei $F = W^\emptyset(\text{fakt})$.

Es ist

$$\begin{aligned} & F(3) \\ = & W^{\{\langle \text{fakt}, F \rangle, \langle n, 3 \rangle\}}(\text{if } n = 0 \text{ then } 1 \text{ else } n * \text{fakt}(n-1)) \\ = & W^{\{\langle \text{fakt}, F \rangle, \langle n, 3 \rangle\}}(n * \text{fakt}(n-1)) \\ = & W^{\{\langle \text{fakt}, F \rangle, \langle n, 3 \rangle\}}(n) \cdot W^{\{\langle \text{fakt}, F \rangle, \langle n, 3 \rangle\}}(\text{fakt}(n-1)) \\ = & 3 \cdot F(W^{\{\langle \text{fakt}, F \rangle, \langle n, 3 \rangle\}}(n - 1)) \\ = & 3 \cdot F(2) \\ = & W^{\{\langle \text{fakt}, F \rangle, \langle n, 2 \rangle\}}(\text{if } n = 0 \text{ then } 1 \text{ else } n * \text{fakt}(n-1)) \\ = & \dots = 6 \end{aligned}$$

Abstrakte Syntax

Die Definition von W bezieht sich auf *abstrakte Syntax*: Wir setzen voraus, dass der äußerste Operator eines verschachtelten Terms bekannt ist. Strenggenommen ist somit W auf *Herleitungsbäumen* (Ausgabe des Parsers) definiert.

Wie sollte man sonst z.B.

$$W^U(2 * 3 - 1)$$

verstehen? Als $W^U(2 * 3) - W^U(1)$ oder $W^U(2) \cdot W^U(3 - 1)$?

Im Interpreter oder Compiler werden tatsächlich die Herleitungsbäume ausgewertet.

Strikte Auswertung

Die Tatsache, dass alle Argumente eines Funktionsaufrufes definiert sein müssen, damit der Funktionsaufruf terminiert, bezeichnet man als *strikte Auswertung*.

In anderen Programmiersprachen gibt es die *verzögerte Auswertung*, wo nicht benutzte Terme auch nicht ausgewertet werden. Z.B.: in Haskell

```
f x y = y + 1
1 x = 1 x + 1
f (1 0) 3
----> 4
```

In Haskell wird zudem ein einmal ausgewerteter Term als Wert abgespeichert und nicht nochmal ausgewertet. Beispiel:

```
u + u where u = f (1 0) 4
```