4.2 Rechnen mit freien Variablen

In diesem Kapitel wollen wir erläutern, wie man überhaupt mit freien Variablen rechnen kann, insbesondere, wie die Implementierung einer logisch-funktionales Programmiersprache die Bindungen für freie Variablen berechnen kann.

Zu Erinnerung: Die funktionale Programmierung basiert auf dem Prinzip der Termersetzung, d.h. gerechnet wird durch *Reduktion* von Ausdrücken, was im Detail bedeutet:

1. Suche den zu reduzierenden Teilausdruck
2. Suche eine „passende“ Regel (Funktionsgleichung)
3. Ersetze Regelvariablen durch „passende“ Ausdrücke (pattern matching)

Formal ist ein Reduktionsschritt so definiert:

$e \to e[\sigma(r)]_p$

falls $l \to r$ eine Regel ist und σ eine Substitution mit $\sigma(l) = e|_p$.

Beispiel: Die Funktion `f` sei wie folgt definiert:

```
f 0 x = 0
f 1 x = x
```

Mittels Reduktion sind die Werte der folgenden Ausdrücke berechenbar:

```
f 0 1 → 0
f 1 2 → 2
f 0 x → 0
```

wobei `x` eine freie Variable ist.

Allerdings können wir mittels Reduktion nicht die Gleichung

```
f z 1 == 1
```

lösen, obwohl theoretisch $\sigma = \{\texttt{z} \mapsto 1\}$ eine Lösung wäre, denn

$\sigma($`f z 1 == 1`$) \;=\;$ `f 1 1 == 1` $\;\to^*\;$ `True`

Um also Ausdrücke, in denen freie Variablen vorkommen, auszurechnen, benötigen wir eine passende Belegung der Variablen, sodass der Ausdruck reduzierbar wird.

Das Problem ist nur: Wie findet man eine passende Belegung?

- Raten von Belegungen: dies ist zu ineffizient, da es im Allgemeinen unendlich viele mögliche Belegungen gibt.
- Eine konstruktive Methode ist, bei der Reduktion das Matching durch Unifikation zu ersetzen. Die führt zu einer Methode, die unter dem Begriff **Narrowing** (Verengen des Lösungsraumes) bekannt ist.

Im Folgenden betrachten wir Termersetzungssysteme als Programme (da Funktionen höherer Ordnung hier erst einmal nicht so relevant sind), d.h. R ist ein vorgegebenes Termersetzungssystem (was dem eingegebenen Programm entspricht).

Definition 4.1 *Wenn t und t' Terme sind, dann heißt*

$t \leadsto_\sigma t'$

Narrowingschritt *(bzgl. R), falls gilt:*

1. *p ist eine nichtvariable Position in t (d.h. $t|_p \notin V$).*
2. *$l \to r$ ist Variante einer Regel aus R (d.h. ersetze in der Regel Variablen durch neue Variablen, sodass Namenskonflikte mit freien Variablen vermieden werden).*
3. *σ ist ein mgu (allgemeinster Unifikator, vgl. Kapitel 2.5.2) für $t|_p$ und l, d.h. es gilt $\sigma(t|_p) = \sigma(l)$.*
4. *$t' = \sigma(t[r]_p)$*

Wir führen einige weitere Notationen für Narrowing-Schritte ein:

- *$t \leadsto_{p,l\to r,\sigma} t'$ falls die Position und/oder die Regel wichtig sind.*
- *$t \leadsto_{\sigma'} t'$ mit*

  $$\sigma'(x) = \begin{cases} \sigma(x) & \text{falls } x \in Var(t) \\ x & \text{sonst} \end{cases}$$

  falls die Belegung der Regelvariablen unwichtig ist. σ' ist also die Einschränkung von σ auf die Variablen im Term t. In diesem Fall schreiben wir dazu auch $\sigma' = \sigma|_{Var(t)}$.
- *$t_0 \leadsto_\sigma^* t_n$ falls*

  $$t_0 \leadsto_{\sigma_1} t_1 \leadsto_{\sigma_2} t_2 \leadsto \cdots \leadsto_{\sigma_n} t_n$$

  und $\sigma = \sigma_n \circ \ldots \circ \sigma_1$
  (hierbei bezeichnet $\circ$ die Komposition von Funktionen)

Für das obige Beispiel gibt es zwei mögliche Narrowing-Schritte:

`f z 1` $\leadsto_{\{\mathtt{z}\mapsto 0\}}$ `0`

`f z 1` $\leadsto_{\{\mathtt{z}\mapsto 1\}}$ `1`

Die Unterschiede zu Reduktion sind somit:

1. Freie Variablen werden durch Unifikation belegt. Hierbei sollte beachtet werden, dass freie Variablen mehrfach vorkommen können, daher wäre in der Bedingung 4. eine Forderung wie $t' = t[\sigma(r)]_p$ **nicht** ausreichend!

2. Im Allgemeinen gibt es mehrere mögliche Belegungen der Variablen, sodass Narrowing-Schritte **nichtdeterministisch** sind.

Beispiel: Wir betrachte noch einmal unser Verwandtschaftsbeispiel aus Kap. 4.1 und die folgende Anfrage: Welche Kinder hat Johann?

```
vater k == Johann
→ ehemann (mutter k) == Johann
⇝{k↦Susanne} ehemann Monika == Johann
→ Johann == Johann
→ True
```

Es ist aber auch folgende Berechnung möglich:

```
vater k == Johann
→ ehemann (mutter k) == Johann
⇝{k↦Peter} ehemann Monika == Johann
→ Johann == Johann
→ True
```

Beide alternativen Berechnungn führen in diesem Fall zu verschiedenen Lösungen. Wenn wir also an allen Lösungen interessiert sind, müssen alle möglichen Narrowing-Schritte durchgeführt werden.

Die Implementierung von Nichtdeterminismus ist aufwändig, sodass es für eine gute Implementierung wichtig ist, möglichst wenig Nichtdeterminismus zu erzeugen. Hierzu sind gute Strategien notwendig, die wir später erläutern.

Eine wichtige Beobachtung ist, dass Narrowing als Verallgemeinerung der Reduktion aufgefasst werden kann. Dies kann man wie folgt sehen:

Wir nehmen an, dass ein Reduktionsschritt möglich ist, d.h. es gilt:

$$t \to t[\sigma(r)]_p$$

mit $\sigma(l) = t|_p$ für eine Regel $l \to r$.

Hierbei sei $\mathcal{D}om(\sigma) \subseteq \mathcal{V}ar(l)$ und $\mathcal{V}ar(l) \cap \mathcal{V}ar(t) = \varnothing$ (dies können wir immer erreichen, da $l \to r$ eine Variante einer Regel sein kann). Dann gilt:

$$\sigma(t) = t$$

und somit ist σ ein mgu für l und $t|_p$. Hieraus folgt:

$$t \leadsto_\sigma \underbrace{\sigma(t[r]_p)}_{=t[\sigma(r)]_p}$$

ist ein Narrowing-Schritt.

Somit gilt:

Jeder Reduktionsschritt ist auch ein Narrowing-Schritt (aber nicht umgekehrt!).

Zusammengefasst ist das Ziel bei der Reduktion die Berechnung einer (bzw. der) Normalform, während das Ziel beim Narrowing das Finden von Werten für Variablen (Variablenbelegungen) ist, sodass eine Normalform berechenbar wird.

Der Ausgangspunkt bei Narrowing-Ableitungen sind in der Regel **Gleichungen**, die zu lösen sind. Formal definieren wir dies wie folgt (hier benutzen wir nicht den Gleichheitsoperator "==", da dessen Bedeutung unterschiedlich ist, wie wir noch sehen werden):

Definition 4.2 *Eine **Gleichung** $s \doteq t$ heißt **gültig** (bzgl. R) falls $s \leftrightarrow_R^* t$ (d.h. s kann in t überführt werden).*

Beispiel:
Addition auf natürlichen Zahlen (hierbei bezeichnet s die Nachfolgerfunktion):

$$\begin{array}{rcl} 0 + n & \to & n \\ s(m) + n & \to & s(m + n) \end{array}$$

Die Gleichung

$$s(0) + s(0) \doteq s(s(0))$$

ist hier gültig, denn $s(0) + s(0) \to s(0 + s(0)) \to s(s(0))$.

Falls R konfluent und terminierend, dann ist $s \leftrightarrow_R^* t$ äquivalent zu der Existenz einer Normalform u mit $s \to^* u$ und $t \to^* u$. Daher reicht es in diesem Fall, die Normalformen beider Seiten zu berechnen, um eine Gleichung zu prüfen.

Mittels Narrowing kann man Gleichungen **lösen**:

Definition 4.3 *Eine Substitution σ heißt **Lösung** der Gleichung $s \doteq t$, falls $\sigma(s) \doteq \sigma(t)$ gültig ist.*

Beispiel: Wir können eine Lösung der Gleichung $z + 1 = 2$ wie folgt berechnen:

$$\begin{array}{lllll} z + s(0) \doteq s(s(0)) & \leadsto_{\{z \mapsto s(m)\}} & s(m + s(0)) & \doteq & s(s(0)) \\ & \leadsto_{\{m \mapsto 0\}} & s(s(0)) & \doteq & s(s(0)) \end{array}$$

Die berechnete Lösung ist dann die Komposition aller Substitutionen eingeschränkt auf die ursprünglichen freie Variablen:

$$\{z \mapsto s(0)\} \quad = \quad (\{z \mapsto s(m)\} \circ \{m \mapsto 0\})|_{\{z\}}$$

Die übliche Forderung an ein Lösungsverfahren ist:

1. **Korrektheit:** Es sollen nur richtige Lösungen berechnet werden.

2. **Vollständigkeit:** Es sollen alle (bzw. Repräsentanten aller) richtigen Lösungen berechnet werden.

Das allgemeine Narrowing-Verfahren ist im folgenden Sinn korrekt und vollständig:

Satz 4.1 ([Hullot 80]) *Sei R TES, sodass $\rightarrow_R$ konfluent und terminierend ist, und $s \doteq t$ eine Gleichung.*

Korrektheit: *Falls $s \doteq t \rightsquigarrow^*_\sigma s' \doteq t'$ und φ ein mgu für s' und t' ist, dann ist $\varphi \circ \sigma$ Lösung von $s \doteq t$, d.h. $\varphi(\sigma(s)) \doteq \varphi(\sigma(t))$ ist gültig.*

Vollständigkeit: *Falls σ' eine Lösung von $s \doteq t$ ist, dann existiert eine Narrowing-Ableitung $s \doteq t \rightsquigarrow^*_\sigma s' \doteq t'$, ein mgu φ für s' und t' und eine Substitution τ, sodass $\sigma'(x) \doteq \tau(\varphi(\sigma(x)))$ gültig ist für alle Variablen $x \in \mathcal{V}ar(s \doteq t)$.*

Das Ergebnis zur Vollständigkeit ist deswegen etwas komplizierter formuliert, weil mittels Narrowing nur Repräsentanten von Lösungen, aber nicht alle Lösungen exakt berechnet werden. Um dies besser zu verstehen, betrachten wir das folgende Beispiel:

$$\begin{array}{rcl} f(a) & \rightarrow & b \\ g & \rightarrow & a \\ i(x) & \rightarrow & x \end{array}$$

1. Betrachten wir die Gleichung

   $f(x) \doteq b$

   Eine Lösung ist: $\sigma' = \{x \mapsto g\}$

   Narrowing berechnet dagegen:

   $f(x) \doteq b \rightsquigarrow_{\{x \mapsto a\}} b \doteq b$

   Die berechnete Lösung ist also: $\sigma = \{x \mapsto a\}$

   Es gilt jedoch: $\sigma(x) \doteq \sigma'(x)$ ist gültig.

2. Betrachten wir die Gleichung:

   $i(z) \doteq z$

   Eine Lösung ist: $\sigma' = \{z \mapsto a\}$

   Narrowing berechnet nur die eine Lösung: $\sigma = \{\}$ (Identität)

   Es gilt jedoch: σ' ist Spezialfall von σ.

Wir können daher festhalten:

Narrowing berechnet allgemeine Repräsentanten aller möglichen Lösungen.

Um diese wirklich alle zu berechnen, ist jedoch Folgendes notwendig (wegen der Existenz einer Ableitung):

Berechne alle möglichen Narrowing-Ableitungen (d.h. rate Position **und** Regel in jedem Schritt!).

Weil dies sehr viele sein können, ist hierzu eine Verbesserung durch spezielle Strategien notwendig ($\leadsto$ später).