Kapitel 8

Untertypen

Der Begriff des *Untertyps*[1] ist essentiell für objekt-orientierte Programmierung, findet sich aber schon in einfachen imperativen Programmiersprachen wie z.B. PASCAL. Dort gibt es zwei Divisionsoperatoren: `div` für Ganzzahlen und “/” für Fließkommazahlen. In unserer Schreibweise:

$$\frac{i : \mathtt{Int} \qquad j : \mathtt{Int}}{i \ \mathtt{div}\ j : \mathtt{Int}} \qquad \frac{a : \mathtt{Float} \qquad b : \mathtt{Int}}{a/b : \mathtt{Float}}$$

PASCAL erlaubt die Anwendung von “/” auch auf Ganzzahlen; der Compiler führt intern eine Konvertierung nach Fließkomma durch. Eine Typherleitung für den Term i/j sieht dann so aus:

$$\frac{\dfrac{i : \mathtt{Int}}{i : \mathtt{Float}}\,\textrm{T-S}{\small\textrm{UB}} \qquad \dfrac{j : \mathtt{Int}}{j : \mathtt{Float}}\,\textrm{T-S}{\small\textrm{UB}}}{i/j : \mathtt{Float}}\,\textrm{T-D}{\small\textrm{IV}}\textrm{F}{\small\textrm{LOAT}}$$

An den Stellen T-Sub wird eine Typkonversion durchgeführt. Die Konversion ist zulässig, weil `Int` als ein Untertyp von `Float` angesehen werden kann, d.h. jede Ganzzahl kann als Fließkommazahl interpretiert werden. Die zentrale Frage im Zusammenhang mit Untertypen ist: Welche Konversionen sind zulässig und sollen vom Typ-Prüfer bzw. Übersetzer automatisch durchgeführt werden?

8.1 Subsumption

Aus der Mathematik ist bekannt, dass $\mathbb{Z} \subseteq \mathbb{R}$. Diese semantische Relation können wir auf der Typebene reflektieren durch die Untertyp-Relation

$$\mathtt{Int} <: \mathtt{Float}$$

[1] auch *Teiltyp*.

Konversion ist möglich von einem Term eines Typs S in dessen Obertyp T, was wir durch die *Subsumptions-Regel* erfassen:

$$\frac{\Gamma \vdash t : S \qquad S <: T}{\Gamma \vdash t : T}\ \text{T-SUB}$$

Prinzip I (Teilmenge) Ein Typ S kann als Untertyp von T aufgefasst werden, wenn die Menge der Bewohner von S als eine Teilmenge der Bewohner von Typ T angesehen werden kann.

Prinzip II (Ersetzung) Ein Typ S kann als Untertyp von T aufgefasst werden, wenn an jeder Stelle, wo ein Term vom Typ T erwartet wird, ein Term vom Typ S eingesetzt werden kann ohne die Typsicherheit zu gefährden.

8.2 Regeln der Untertyp-Beziehung

[Hier fehlt die Motivation der einzelnen Typregeln]

Definition 8.1 (Deklarative Untertyp-Beziehung) *Die Relation $S <: T$ ist gegeben durch die folgenden Inferenzregeln:*

$$\frac{}{T <: T}\ \text{S-REFL} \qquad \frac{R <: S \qquad S <: T}{R <: T}\ \text{S-TRANS}$$

$$\frac{T_1 <: S_1 \qquad S_2 <: T_2}{S_1 \to S_2 <: T_1 \to T_2}\ \text{S-ARROW} \qquad \frac{S_1 <: T_1 \qquad S_2 <: T_2}{S_1 \times S_2 <: T_1 \times T_2}\ \text{S-PROD}$$

$$\frac{}{\{\overrightarrow{l : T}, \overrightarrow{k : R}\} <: \{\overrightarrow{l : T}\}}\ \text{S-RWIDTH} \qquad \frac{S_i <: T_i \quad \textit{für } i = 1..|\,\overrightarrow{l}\,|}{\{\overrightarrow{l : S}\} <: \{\overrightarrow{l : T}\}}\ \text{S-RDEPTH}$$

8.3 Entscheidung der Untertyp-Beziehung

Satz 8.1 *Die Untertyp-Relation ist entscheidbar.*

Zum Beweis dieses Satzes müssen wir einen Algorithmus angeben, der für beliebige Typen S und T entscheidet ob $S <: T$.

Aus den bisherigen Untertyp-Regeln lässt sich nicht direkt ein Algorithmus ableiten. Insbesondere die Transitivitäts-Regel bereitet Kopfzerbrechen, denn sie ist nicht Syntax-gerichtet. Um zu entscheiden ob $R <: T$, empfiehlt uns die Transitivitäts-Regel, einen Typ S zu finden und dann $R <: S$ und $S <: T$ zu testen. Da es unendlich viele Typen S gibt, kann unsere Suche endlos dauern.

Wir entwickeln im folgenden neue Regeln für Untertypen, die ohne eine Transitivitäts-Regel auskommen, deterministisch und Syntax-gerichtet sind.

Definition 8.2 (Algorithmische Untertyp-Beziehung) *Die Relation $S \vartriangleleft: T$ ist gegeben durch die folgenden Inferenzregeln:*

$$\frac{}{A \vartriangleleft: A}\ \textsc{SA-Refl} \quad \textit{für Grundtypen } A$$

$$\frac{T_1 \vartriangleleft: S_1 \qquad S_2 \vartriangleleft: T_2}{S_1 \to S_2 \vartriangleleft: T_1 \to T_2}\ \textsc{SA-Arrow} \qquad \frac{S_1 \vartriangleleft: T_1 \qquad S_2 \vartriangleleft: T_2}{S_1 \times S_2 \vartriangleleft: T_1 \times T_2}\ \textsc{SA-Prod}$$

$$\frac{S_i \vartriangleleft: T_i \quad \textit{für } i = 1..|\overrightarrow{l}\,|}{\{\overrightarrow{l : S}, \overrightarrow{k : R}\} \vartriangleleft: \{\overrightarrow{l : T}\}}\ \textsc{SA-Record}$$

Aus diesen Regeln lässt sich nun direkt ein Algorithmus ablesen. Bleibt zu zeigen, dass die beiden Untertyp-Beziehungen äquivalent sind. Die eine Richtung ist einfach: Jede Untertyp-Beziehung $S \vartriangleleft: T$, die mit den algorithmischen Regeln hergeleitet wurde, kann auch mit den deklarativen gezeigt werden. Hierfür muss man nur einsehen, dass sich jede Anwendung der Regel SA-Record durch eine Kombination von S-RWidth, S-RDepth, S-Trans und S-Refl ausdrücken lässt.

Lemma 8.1 (Algorithmische Untertyp-Relation ist korrekt) *Wenn $S \vartriangleleft: T$, dann $S <: T$.*

Beweis. Induktion nach $S \vartriangleleft: T$. □

Die andere Richtung ist etwas schwieriger. Hier machen Anwendungen der Regeln S-Refl und S-Trans Umstände. Wir zeigen also zuerst, dass die algorithmische Relation reflexiv und transitiv ist. Die Beweise dieser Aussagen geben uns Algorithmen, wie wir Vorkommen von S-Refl und S-Trans eliminieren können.

Lemma 8.2 (Reflexivität) *Für alle Typen T gilt $T \vartriangleleft: T$.*

Beweis. Durch Induktion nach T. Wir demonstrieren einen Fall:

Fall $T = T_1 \to T_2$. Nach Induktionsvoraussetzung gilt $T_1 \vartriangleleft: T_1$ und $T_2 \vartriangleleft: T_2$. Mit Regel SA-Arrow folgt damit $T_1 \to T_2 \vartriangleleft: T_1 \to T_2$. □

Lemma 8.3 (Transitivität) *Wenn $R \vartriangleleft: S$ und $S \vartriangleleft: T$, dann $R \vartriangleleft: T$.*

Beweis. Durch Induktion nach $R \vartriangleleft: S$ mit Fallunterscheidung über $S \vartriangleleft: T$. Zum Beispiel:

Fall $R = R_1 \to R_2$, $S = S_1 \to S_2$ und

$$\frac{S_1 \vartriangleleft: R_1 \qquad R_2 \vartriangleleft: S_2}{R_1 \to R_2 \vartriangleleft: S_1 \to S_2}\textsc{SA-Arrow} \qquad S_1 \to S_2 \vartriangleleft: T$$

Die einzige Möglichkeit ist $T = T_1 \to T_2$ mit $T_1 \vartriangleleft: S_1$ und $S_2 \vartriangleleft: T_2$. Induktionsvoraussetzung liefert $T_1 \vartriangleleft: R_1$ und $R_2 \vartriangleleft: T_2$. Mit Regel SA-Arrow folgt also $R_1 \to R_2 \vartriangleleft: T_1 \to T_2$. □

Lemma 8.4 (Algorithmische Untertyp-Relation ist vollständig) *Wenn $S <: T$, dann $S <: T$.*

Beweis. Durch Induktion nach $S <: T$, unter Verwendung der Lemmata über Reflexivität und Transitivität. □

Satz 8.1 folgt nun leicht. Die algorithmische Untertyp-Relation lässt sich direkt in ein Programm umwandeln, dass für gegebene S, T entscheidet, ob $S \mathrel{\triangleleft:} T$. Nach Lemmata 8.1 und 8.4 ist dies äquivalent zu $S <: T$.