Erweiterung der Syntax:

Wir betrachten zusätzlich Ausdrücke der Form:

$$\begin{array}{rcl} e & ::= & \ldots \mid [\,] \mid e :: e \mid \textbf{case}\ e_0\ \textbf{of}\ [\,]\ :\ e_1 \mid z :: z\ :\ e_2 \\ & & \mid (e_1, e_2) \mid \textbf{case}\ e_0\ \textbf{of}\ (x_1, x_2)\ :\ e_1 \end{array}$$

Top-Strictness

- Wir nehmen an, das Programm sei korrekt getypt.
- Wir interessieren uns nur für den obersten Konstruktor.
- Wieder modellieren wir diese Eigenschaft durch (monotone) Boolesche Funktionen.
- Für **int**-Werte stimmt dies mit Striktheit überein :-)
- Wir erweitern die abstrakte Auswertung $[\![e]\!]^\sharp\ \rho$ um Regeln für Fallunterscheidungen ...

$$
\begin{array}{lcl}
[\![\textbf{case}\ e_0\ \textbf{of}\ [\,]\ :\ e_1\ \mid\ ::z\ :\ e_2]\!]^\sharp\,\rho & = & [\![e_0]\!]^\sharp\,\rho \wedge ([\![e_1]\!]^\sharp\,\rho \\
 & & \qquad \vee\ [\![e_2]\!]^\sharp\,(\rho \oplus \{z \mapsto 1\}))\\
[\![\textbf{case}\ e_0\ \textbf{of}\ (x_1, x_2)\ :\ e_1]\!]^\sharp & = & [\![e_0]\!]^\sharp\,\rho \wedge [\![e_1]\!]^\sharp\,(\rho \oplus \{x_1, x_2 \mapsto 1\})\\
[\![[\,]]\!]^\sharp\,\rho\ =\ [\![::e]\!]^\sharp\,\rho\ =\ [\![(e_1, e_2)]\!]^\sharp\,\rho & = & 1
\end{array}
$$

- Die Regeln für **case** sehen analog denjenigen für **if** aus.
- Im ::-Fall können wir nichts über die Werte unterhalb des Konstruktors wissen, deshalb $\{z \mapsto 1\}$.
- Wir testen unsere Analyse an der Funktion app ...

Beispiel:

$$\begin{array}{rcl} \mathsf{app} & = & \mathbf{fn}\ x\ \Rightarrow\ \mathbf{fn}\ y\ \Rightarrow\ \mathbf{case}\ x\ \mathbf{of}\ [\,]\ :\ y \\ & & \quad \mid\ ::z\ :\ \mathbf{case}\ z\ \mathbf{of}\ (x, xs)\ :: (x, \mathsf{app}\ xs\ y) \end{array}$$

Abstrakte Interpretation liefert das Gleichungssystem:

$$\begin{array}{rcl} [\![\mathsf{app}]\!]^\sharp\ b_1\ b_2 & = & b_1 \wedge (b_2 \vee 1) \\ & = & b_1 \end{array}$$

Wir schließen, dass wir nur für das erste Argument mit Sicherheit annehmen können, dass sein Top-Konstruktor benötigt wird :-)

Total Strictness

Nehmen wir an, das Ergebnis einer Funktionsanwendung werde ganz benötigt. Welche Argumente werden dann ganz benötigt ?

Wieder verwenden wir Boolesche Funktionen ...

$$
\begin{array}{lcl}
[\![\textbf{case}\ e_0\ \textbf{of}\ [\,]\ :\ e_1\ \mid\ ::z\ :\ e_2]\!]^\sharp\,\rho & = & [\![e_0]\!]^\sharp\,\rho \wedge [\![e_1]\!]^\sharp\,\rho \\
 & & \vee\ [\![e_2]\!]^\sharp\,(\rho \oplus \{z \mapsto [\![e_0]\!]^\sharp\,\rho\}) \\
[\![\textbf{case}\ e_0\ \textbf{of}\ (x_1,x_2)\ :\ e_1]\!]^\sharp\,\rho & = & \textbf{let}\ b = [\![e_0]\!]^\sharp\,\rho \\
 & & \textbf{in}\ [\![e_1]\!]^\sharp\,(\rho \oplus \{x_1 \mapsto 1, x_2 \mapsto b\}) \vee \\
 & & \quad [\![e_1]\!]^\sharp\,(\rho \oplus \{x_1 \mapsto b, x_2 \mapsto 1\}) \\
[\![[\,]]\!]^\sharp\,\rho & = & 1 \\
[\![::e]\!]^\sharp\,\rho & = & [\![e]\!]^\sharp\,\rho \\
[\![(e_1,e_2)]\!]^\sharp\,\rho & = & [\![e_1]\!]^\sharp\,\rho \wedge [\![e_2]\!]^\sharp\,\rho
\end{array}
$$

Diskussion:

- Die Regeln f ür Konstruktoranwendungen haben sich geändert.
- Auch berücksichtigt die Behandlung von **case** nun die Bestandteile z bzw. x_1, x_2.
- Wieder testen wir den Ansatz für die Funktion app.

Beispiel:

Abstrakte Interpretation liefert das Gleichungssystem:

$$\begin{aligned}
[\![\mathsf{app}]\!]^\sharp\ b_1\ b_2 &= b_1 \wedge b_2 \vee b_1 \wedge [\![\mathsf{app}]\!]^\sharp\ 1\ b_2 \vee 1 \wedge [\![\mathsf{app}]\!]^\sharp\ b_1\ b_2 \\
&= b_1 \wedge b_2 \vee b_1 \wedge [\![\mathsf{app}]\!]^\sharp\ 1\ b_2 \vee [\![\mathsf{app}]\!]^\sharp\ b_1\ b_2
\end{aligned}$$

Das liefert die folgende Fixpunktiteration:

0	$\mathbf{fn}\, x \Rightarrow \mathbf{fn}\, y \Rightarrow 0$
1	$\mathbf{fn}\, x \Rightarrow \mathbf{fn}\, y \Rightarrow x \wedge y$
2	$\mathbf{fn}\, x \Rightarrow \mathbf{fn}\, y \Rightarrow x \wedge y$

Wir schließen, dass wir beide Argumente mit Sicherheit ganz benötigen, falls das Ergebnis ganz benötigt wird :-)

Achtung:

Ob das Ergebnis ganz benötigt wird, hängt vom Kontext des Funktionsaufrufs ab!

Für eine solche Umgebung kann man eine spezialisierte Funktion aufrufen ...

$$
\begin{array}{lcl}
\mathsf{app\#} & = & \mathbf{fn}\ x\ \Rightarrow\ \mathbf{fn}\ y\ \Rightarrow\ \mathbf{case}\ x\ \mathbf{of}\ [\,]\ :\ \mathbf{let}\ \#\,y' = y\ \mathbf{in}\ y' \\
& & \quad \mid\ ::z\ :\ \ \mathbf{case}\ z\ \mathbf{of}\ (x, xs)\ : \\
& & \qquad\qquad \mathbf{let}\ \#\,r = ::\,(x, \mathsf{app\#}\ xs\ y) \\
& & \qquad\qquad \mathbf{in}\ r
\end{array}
$$

Diskussion:

- Beide Arten von Striktheitsanalyse verwenden den gleichen Verband.
- Ergebnisse und Verwendung sind jedoch unterschiedlich :-)
- Dabei verwenden wir die Beschreibungsrelationen:

  Top Strictness : $\bot\ \Delta\ 0$

  Total Strictness : $z\ \Delta\ 0$ sofern $\bot$ in z vorkommt.
- Beide Analysen lassen sich zu einer Analyse kombinieren ...

Kombinierte Striktheitsanalyse

- Wir verwenden den Verband:

$$\mathbb{T} = \{0 \sqsubset 1 \sqsubset 2\}$$

- Die Beschreibungsrelation ist gegeben durch:

$$\bot \; \Delta \; 0 \quad z \; \Delta \; 1 \; (z \text{ enthält } \bot) \quad z \; \Delta \; 2 \; (z \text{ Wert})$$

- Der Verband ist informativer, Funktionen aber leider nicht mehr so effizient beschreibbar z.B. durch Boolesche Ausdrücke :-(
- Wir benötigen die Hilfsfunktionen:

$$(i \sqsubseteq x);\; y = \begin{cases} y & \text{falls } i \sqsubseteq x \\ 0 & \text{sonst} \end{cases}$$

Die kombinierte Auswertungsfunktion:

$$
\begin{array}{lcl}
[\![\textbf{case}\, e_0\, \textbf{of}\, [\,] : e_1 \mid :: z : e_2]\!]^\sharp\, \rho & = & (2 \sqsubseteq [\![e_0]\!]^\sharp\, \rho)\,;\, [\![e_1]\!]^\sharp\, \rho \sqcup \\
 & & (1 \sqsubseteq [\![e_0]\!]^\sharp\, \rho)\,;\, [\![e_2]\!]^\sharp\, (\rho \oplus \{z \mapsto [\![e_0]\!]^\sharp\, \rho\}) \\
[\![\textbf{case}\, e_0\, \textbf{of}\, (x_1, x_2) : e_1]\!]^\sharp\, \rho & = & \textbf{let}\ b = [\![e_0]\!]^\sharp\, \rho \\
 & & \textbf{in}\ (1 \sqsubseteq [\![e_0]\!]^\sharp\, \rho)\,; \\
 & & \quad ([\![e_1]\!]^\sharp\, (\rho \oplus \{x_1 \mapsto 2, x_2 \mapsto b\}) \sqcup \\
 & & \quad [\![e_1]\!]^\sharp\, (\rho \oplus \{x_1 \mapsto b, x_2 \mapsto 2\})) \\
[\![[\,]]\!]^\sharp\, \rho & = & 2 \\
[\![:: e]\!]^\sharp\, \rho & = & 1 \sqcup [\![e]\!]^\sharp\, \rho \\
[\![(e_1, e_2)]\!]^\sharp\, \rho & = & 1 \sqcup ([\![e_1]\!]^\sharp\, \rho \sqcap [\![e_2]\!]^\sharp\, \rho)
\end{array}
$$

Beispiel:

Für unsere Lieblingsfunktion app erhalten wir:

$$
\begin{array}{rcl}
[\![\mathsf{app}]\!]^\sharp\ d_1\ d_2 & = & (2 \sqsubseteq d_1)\,;\, d_2 \sqcup \\
 & & (1 \sqsubseteq d_1)\,;\, (1 \sqcup [\![\mathsf{app}]\!]^\sharp\ d_1\ d_2 \sqcup d_1 \sqcap [\![\mathsf{app}]\!]^\sharp\ 2\ d_2) \\
 & = & (2 \sqsubseteq d_1)\,;\, d_2 \sqcup \\
 & & (1 \sqsubseteq d_1)\,;\, 1 \sqcup \\
 & & (1 \sqsubseteq d_1)\,;\, [\![\mathsf{app}]\!]^\sharp\ d_1\ d_2 \sqcup \\
 & & d_1 \sqcap [\![\mathsf{app}]\!]^\sharp\ 2\ d_2
\end{array}
$$

Das liefert die folgende Fixpunktiteration:

0	$\mathbf{fn}\, x \Rightarrow \mathbf{fn}\, y \Rightarrow\ 0$
1	$\mathbf{fn}\, x \Rightarrow \mathbf{fn}\, y \Rightarrow\ (2 \sqsubseteq x);\ y \sqcup (1 \sqsubseteq x);\ 1$
2	$\mathbf{fn}\, x \Rightarrow \mathbf{fn}\, y \Rightarrow\ (2 \sqsubseteq x);\ y \sqcup (1 \sqsubseteq x);\ 1$

Wir schließen,

- dass wir beide Argumente ganz benötigen, falls das Ergebnis ganz benötigt wird, und
- dass wir die Wurzel des ersten Argument brauchen, wenn wir die Wurzel des Ergebnisses brauchen :-)

Bemerkung:

Unsere Analyse lässt sich leicht verallgemeinern, um Ausgewertetheit bis zu einer Tiefe d zu garantieren ;-)

Ausblick:

- Unsere Ansätze sind auch für andere Datenstrukturen anwendbar.
- Prinzipiell können wir so auch höhere (monomorphe) Funktionen analysieren :-)
- Dann benötigen wir jedoch höhere abstrakte Funktionen — davon gibt es leider sehr viele :-(
- Solche Funktionen werden darum durch

$$\textbf{fn}\ x_1 \Rightarrow \ldots\ \textbf{fn}\ x_r \Rightarrow \top$$

  approximiert :-)
- Für einige bekannte höhere Funktionen wie map, foldl, loop, ... kann man diesen Ansatz noch verbessern :-))