Lokale Definitionen:

$$
\begin{array}{l}
(e_1, \eta) \Longrightarrow (v_1, \eta_1) \\
(e_2, \eta \oplus \{x_1 \mapsto (v_1, \eta_1)\}) \Longrightarrow (v_2, \eta_2) \\
\ldots \\
(e_k, \eta \oplus \{x_1 \mapsto (v_1, \eta_1), \ldots, x_{k-1} \mapsto (v_{k-1}, \eta_{k-1})\}) \Longrightarrow (v_k, \eta_k) \\
(e_0, \eta \oplus \{x_1 \mapsto (v_1, \eta_1), \ldots, x_k \mapsto (v_k, \eta_k)\}) \Longrightarrow (v_0, \eta_0)
\end{array}
$$

$$
(\textbf{let } x_1 = e_1 \ldots x_k = e_k \textbf{ in } e_0 \textbf{ end}, \eta) \Longrightarrow (v_0, \eta_0)
$$

Variablen:

$$\frac{\eta\,(x) \;=\; (v_1, \eta_1)}{(x, \eta) \Longrightarrow (v_1, \eta_1)}$$

Korrektheit der Analyse:

Man zeigt für jedes (e, η), das in einer Ableitung für das Programm vorkommt:

- Falls $\eta(x) = (v, \eta_1)$, dann ist $v \in [\![x]\!]^\sharp$.
- Falls $(e, \eta) \Longrightarrow (v, \eta_1)$, dann ist $v \in [\![e]\!]^\sharp$.

Fazit:

$[\![e]\!]^\sharp$ liefert eine Obermenge der Werte, zu denen sich e möglicherweise ausrechnet :-)

4.4 Anwendung: Inlining

Probleme:

- globale Variablen. Das Programm:

$$
\begin{array}{lll}
\textbf{let} & x = 1 & \\
 & f = & \textbf{let} \quad x = 2 \\
 & & \textbf{in} \quad \textbf{fn}\ y \Rightarrow y + x \\
 & & \textbf{end} \\
\textbf{in} & f\ x & \\
\textbf{end} & &
\end{array}
$$

... berechnet offenbar etwas anderes als:

$$
\begin{array}{lll}
\textbf{let} & x = 1 & \\
 & f = & \textbf{let} \quad x = 2 \\
 & & \textbf{in} \quad \textbf{fn}\ y \ \Rightarrow\ y + x \\
 & & \textbf{end} \\
\textbf{in} & \boxed{\begin{array}{ll} \textbf{let} & y = x \\ \textbf{in} & y + x \\ \textbf{end} & \end{array}} & \\
\textbf{end} & &
\end{array}
$$

- rekursive Funktionen. In der Definition:

$$x = \mathsf{fix}\ (\mathsf{foo},\ \textbf{fn}\ y \ \Rightarrow\ \mathsf{foo}\ y)$$

sollten wir foo besser nicht substituieren :-)

Idee 1:

→ Wir machen erstmal die Namen im Programm eindeutig.

→ Dann substituieren wir nur Funktionen, die statisch im Scope der selben globalen Variablen stehn, wie die Anwendung :-)

→ Wir berechnen für jeden Ausdruck alle Funktions-Definitionen mit dieser Eigenschaft :-)

Sei $D[e]$ die Menge der Definitionen, die in e statisch ankommen.

- Für $e \equiv \textbf{let}\ x_1 = e_1 \ldots x_k = e_k\ \textbf{in}\ e_0\ \textbf{end}$ haben wir:

$$\begin{aligned} D[e_1] &= D \\ &\ldots \\ D[e_k] &= D \cup \{x_1, \ldots, x_{k-1}\} \\ D[e_0] &= D \cup \{x_1, \ldots, x_k\} \end{aligned}$$

- In den anderen Fällen propagiert sich D unverändert zu den Teilausdrücken :-)

  Für $e \equiv \textbf{fn}\ x \Rightarrow e_1$ haben wir etwa:

$$D[e_1] = D$$

... im Beispiel:

$$
\begin{array}{lll}
\textbf{let} & x = 1 & \\
 & f = & \textbf{let} \quad x_1 = 2 \\
 & & \textbf{in} \quad \textbf{fn}\ y \Rightarrow y + x_1 \\
 & & \textbf{end} \\
\textbf{in} & f\ x & \\
\textbf{end} & &
\end{array}
$$

... steht (nach Umbenennung :-) f für $f\ x$ statisch zur Verfügung. Bei Substitution erhalten wir:

$$
\begin{array}{lll}
\textbf{let} & x = 1 & \\
 & f = & \textbf{let} \quad x_1 = 2 \\
 & & \textbf{in} \quad \textbf{fn}\ y \Rightarrow y + x_1 \\
 & & \textbf{end} \\
\textbf{in} & \boxed{\begin{array}{l}
\textbf{let} \quad y = x \\
\textbf{in} \quad \textbf{let} \quad x_1 = 2 \\
\qquad\quad \textbf{in} \quad y + x_1 \\
\qquad\quad \textbf{end} \\
\textbf{end}
\end{array}} & \\
\textbf{end} & &
\end{array}
$$

Ersetzen der Variablen-Variablen-Umbenennungen ergibt schließlich:

$$
\begin{array}{lll}
\textbf{let} & x = 1 & \\
 & f = & \textbf{let} \quad x_1 = 2 \\
 & & \textbf{in} \quad \textbf{fn}\ y \Rightarrow y + x_1 \\
 & & \textbf{end} \\
\textbf{in} & \boxed{\begin{array}{ll} \textbf{let} & x_1 = 2 \\ \textbf{in} & x + x_1 \\ \textbf{end} & \end{array}} & \\
\textbf{end} & &
\end{array}
$$

Idee 2:

→ Wir benutzen unsere Wert-Analyse.

→ Wir ignorieren globale Variablen :-)

→ Wir substituieren nur Funktionen ohne freie Variablen :-))

Beispiel: Die map-Funktion

$$
\begin{array}{l}
\textbf{let}\ f = \textbf{fn}\ x \ \Rightarrow\ x \cdot x \\
\quad \mathsf{map} = \mathsf{fix}\,(\mathsf{map},\ \textbf{fn}\ g \ \Rightarrow\ \textbf{fn}\ x \ \Rightarrow\ \textbf{case}\ x \\
\qquad\qquad\qquad\qquad \textbf{of}\ \ [\,]\ :\ \ [\,] \\
\qquad\qquad\qquad\qquad\ \ \mid\ \ ::z:\ \ \textbf{case}\ z\ \textbf{of}\ (x_1, x_2)\ \textbf{in} \\
\qquad\qquad\qquad\qquad\qquad\qquad \textbf{let}\ y_1 = g\ x_1 \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad m = \mathsf{map}\ g \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad y_2 = m\ x_2 \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad z_1 = (y_1, y_2) \\
\qquad\qquad\qquad\qquad\qquad\qquad \textbf{in}\ ::z_1 \\
\qquad\qquad\qquad\qquad\qquad\qquad \textbf{end} \\
\qquad\qquad\qquad\qquad \textbf{end}) \\
\quad h = \mathsf{map}\ \ f \\
\textbf{in}\ \ h\ \mathit{list} \\
\textbf{end}
\end{array}
$$

- Der formale Parameter g von map ist stets f :-)
- Wir können die Anwendung von f in der Definition von map ersetzen:

$$
\begin{array}{l}
\mathsf{map} = \mathsf{fix}\,(\mathsf{map},\ \mathbf{fn}\ g\ \Rightarrow\ \mathbf{fn}\ x\ \Rightarrow\ \mathbf{case}\ x \\
\qquad\qquad \mathbf{of}\ \ [\,]\ :\ \ [\,] \\
\qquad\qquad\quad \mid\ \ ::\ z\ :\ \ \mathbf{case}\ z\ \mathbf{of}\ (x_1, x_2)\ \mathbf{in} \\
\qquad\qquad\qquad\qquad \mathbf{let}\ y_1 = \boxed{\begin{array}{l}\mathbf{let}\ x = x_1 \\ \mathbf{in}\ x \cdot x \\ \mathbf{end}\end{array}} \\
\qquad\qquad\qquad\qquad\qquad m = \mathsf{map}\ g \\
\qquad\qquad\qquad\qquad\qquad y_2 = m\ x_2 \\
\qquad\qquad\qquad\qquad\qquad z_1 = (y_1, y_2) \\
\qquad\qquad\qquad\qquad \mathbf{in}\ \ ::\ z_1 \\
\qquad\qquad\qquad\qquad \mathbf{end} \\
\qquad\qquad \mathbf{end}) \\
h = \mathsf{map}\ \ f
\end{array}
$$

- Noch mehr könnten wir sparen, wenn wir die spezialisierte Funktion $h = \mathsf{map}\ f$ direkt definieren könnten :-)
- Dazu müssen wir überall in der Definition von map das Muster $\boxed{\mathsf{map}\ g}$ durch h ersetzen ...

$\Longrightarrow$ fold-Transformation :-)

- Alle weiteren Vorkommen von g müssen durch (die Definition von) f ersetzt werden ...

  // kommt hier nicht vor :-)

$$
\begin{array}{l}
\mathsf{map} = \mathsf{fix}\,(\boxed{\mathsf{map},\ \mathbf{fn}\ g\ \Rightarrow}\ \mathbf{fn}\ x\ \Rightarrow\ \mathbf{case}\ x \\
\qquad\qquad \mathbf{of}\ \ [\,]\ :\ \ [\,] \\
\qquad\qquad\quad \mid\ \ ::\ z\ :\ \ \mathbf{case}\ z\ \mathbf{of}\ (x_1, x_2)\ \mathbf{in} \\
\qquad\qquad\qquad\qquad \mathbf{let}\ y_1 = \boxed{\begin{array}{l}\mathbf{let}\ x = x_1 \\ \mathbf{in}\ x \cdot x \\ \mathbf{end}\end{array}} \\
\qquad\qquad\qquad\qquad\qquad\quad m = \boxed{\mathsf{map}\ g} \\
\qquad\qquad\qquad\qquad\qquad\quad y_2 = m\ x_2 \\
\qquad\qquad\qquad\qquad\qquad\quad z_1 = (y_1, y_2) \\
\qquad\qquad\qquad\qquad \mathbf{in}\ \ ::\ z_1 \\
\qquad\qquad\qquad\qquad \mathbf{end} \\
\qquad\qquad \mathbf{end}) \\
h = \mathsf{map}\ \ f
\end{array}
$$

$$
\begin{array}{l}
h = \mathsf{fix}\,(h, \mathbf{fn}\ x \ \Rightarrow\ \mathbf{case}\ x \\
\qquad\qquad \mathbf{of}\ \ [\,]\ :\ \ [\,] \\
\qquad\qquad\ \ \mid\ \ ::\ z\ :\ \ \mathbf{case}\ z\ \mathbf{of}\ (x_1, x_2)\ \mathbf{in} \\
\qquad\qquad\qquad\qquad \mathbf{let}\ y_1 = \boxed{\begin{array}{l}\mathbf{let}\ x = x_1 \\ \mathbf{in}\ x \cdot x \\ \mathbf{end}\end{array}} \\
\qquad\qquad\qquad\qquad\qquad\quad m = \boxed{h} \\
\qquad\qquad\qquad\qquad\qquad\quad y_2 = m\ x_2 \\
\qquad\qquad\qquad\qquad\qquad\quad z_1 = (y_1, y_2) \\
\qquad\qquad\qquad\qquad \mathbf{in}\ \ ::\ z_1 \\
\qquad\qquad\qquad\qquad \mathbf{end} \\
\qquad\qquad \mathbf{end})
\end{array}
$$