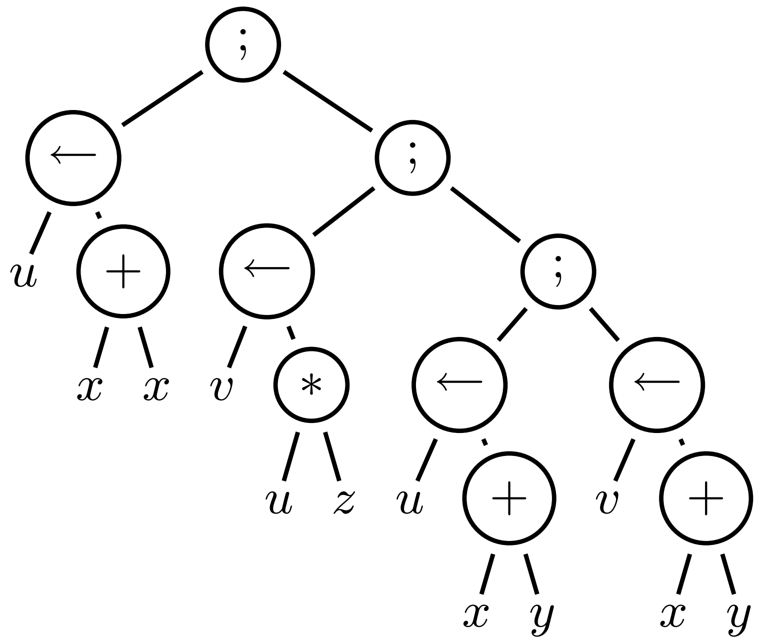
# SLC–Beispielprogramm

$$\pi = (IV, \beta, OV) \in SLC \quad \text{mit} \quad \begin{array}{rl} IV : & x, y, z; \\ \beta : & u \leftarrow x + x; \\ & v \leftarrow u * z; \\ & u \leftarrow x + y; \\ & v \leftarrow x + y; \\ OV : & u, v \end{array}$$

Flussdiagramm von $\pi$:

$$\begin{array}{c} \boxed{x, y, z \leftarrow input} \\ \downarrow \\ \boxed{u \leftarrow x + x} \\ \downarrow \\ \boxed{v \leftarrow u * z} \\ \downarrow \\ \boxed{u \leftarrow x + y} \\ \downarrow \\ \boxed{v \leftarrow x + y} \\ \downarrow \\ \boxed{output \leftarrow u, v} \end{array}$$

AST von $\beta$:



# Beispiel: Common Subexpression Elimination

Ausgangsprogramm:

$$
\begin{array}{rcll}
\pi & = & (IV, \beta, OV) \in SLC \text{ mit} & \\
IV & : & x, y; & \\
\beta & : & z \leftarrow x + y; & (1) \\
 & & z \leftarrow z * y; & \\
 & & x \leftarrow x + y; & (2) \\
 & & v \leftarrow x + y; & (3) \\
 & & w \leftarrow v - z; & \\
 & & w \leftarrow w + v; & \\
OV & : & w &
\end{array}
$$

Einführung einer temporären Variablen $tv_1$:

$$
\begin{array}{rcl}
\multicolumn{3}{l}{T_{cs}(\pi) = (IV, \beta', OV) \in SLC \text{ mit}} \\
IV & : & x, y; \\
\beta' & : & tv_1 \leftarrow x + y; \\
 & & z \leftarrow tv_1; \\
 & & z \leftarrow z * y; \\
 & & x \leftarrow tv_1; \\
 & & v \leftarrow x + y; \\
 & & w \leftarrow v - z; \\
 & & w \leftarrow w + v; \\
OV & : & w
\end{array}
$$

Beachte: $x + y$ (1) ist bei (2) verfügbar, aber nicht bei (3).

# Live Variables–Analyse und Dead Code Elimination

$$\begin{array}{rcl} \pi & = & (IV, \beta, OV) \\ IV & : & x, y, z; \\ \beta & : & u \leftarrow x + x; \quad (*) \\ & & v \leftarrow u * z; \quad (*) \\ & & u \leftarrow x + y; \\ & & v \leftarrow x + y; \\ OV & : & u, v \end{array} \qquad \begin{array}{rcl} \pi' & = & (IV, \beta', OV) \\ IV & : & x, y, z; \\ \beta' & : & u \leftarrow x + y; \\ & & v \leftarrow x + y; \\ OV & : & u, v \end{array}$$

Die Anweisungen mit $(*)$ sind überflüssig („*dead code*“) und können eliminiert werden.

Es folgt: $\pi \sim \pi'$ und $c(\pi') < c(\pi)$ bzgl. Codelänge

# Reaching Definitions–Analyse und Konstantenfaltung

$$\begin{aligned}
\pi &= (IV, \beta, OV) \\
IV &: x; \\
\beta &: y \leftarrow 10; \\
&\quad z \leftarrow 2; \\
&\quad z \leftarrow y * z; \\
&\quad y \leftarrow y + x; \\
&\quad z \leftarrow y * z; \\
OV &: z
\end{aligned}$$

$$\begin{aligned}
\pi' &= (IV, \beta', OV) \\
IV &: x; \\
\beta' &: y \leftarrow 10; \\
&\quad z \leftarrow 2; \\
&\quad z \leftarrow 20; \\
&\quad y \leftarrow 10 + x; \\
&\quad z \leftarrow y * 20; \\
OV &: z
\end{aligned}$$

Interpretation: $< \mathbb{Z}; +, *, 0, 1, -1, \ldots >$

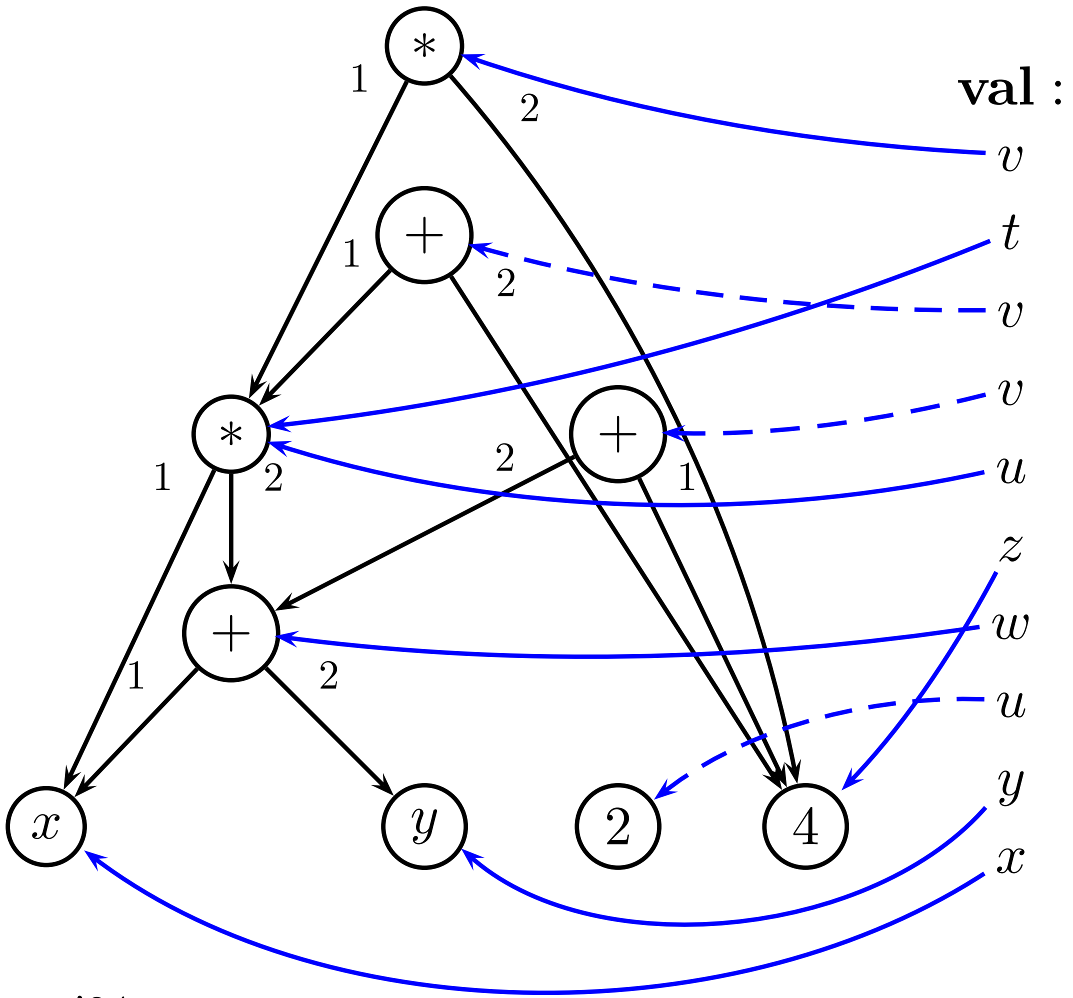
# Beispiel: DAG–Konstruktion

Ausgangsprogramm:

$$\pi = (IV, \beta, OV) \in SLC$$

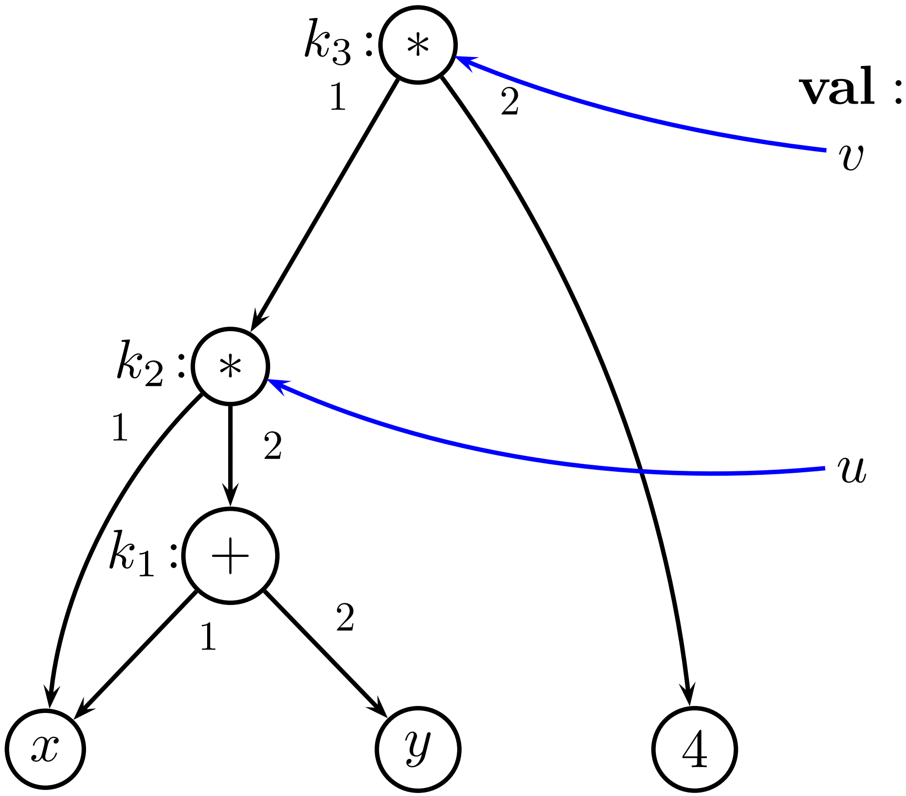
$$\begin{array}{rcl}
IV & : & x, y; \\
\beta & : & u \leftarrow 2; \\
 & & w \leftarrow x + y; \\
 & & z \leftarrow u + 2; \\
 & & u \leftarrow x * w; \\
 & & v \leftarrow 4 + w; \\
 & & v \leftarrow u + z; \\
 & & t \leftarrow x * w; \\
 & & v \leftarrow t * z; \\
OV & : & u, v
\end{array}$$

DAG–Darstellung $D(IV, \beta)$:



# Beispiel: DAG–Konstruktion

DAG–Darstellung $D(\pi)$:




$$
\begin{aligned}
k_1 &\leftarrow x + y;\\
k_2 &\leftarrow x * k_1;\\
k_3 &\leftarrow k_2 * 4;
\end{aligned}
$$

$$\pi_D = (IV, \beta_D, OV) \in SLC$$

$$
\begin{aligned}
\beta_D :\ k_1 &\leftarrow x + y;\\
u &\leftarrow x * k_1;\\
v &\leftarrow u * 4;
\end{aligned}
$$

# Optimierung von Polynomen

$$f(x) = ax^3 + bx^2 + cx + d$$

$$\pi = (IV, \beta, OV) \in SLC$$

$$\begin{array}{rcl} IV & : & x; \\ \beta & : & u \leftarrow a * x; \\ & & u \leftarrow u * x; \\ & & u \leftarrow u * x; \\ & & v \leftarrow b * x; \\ & & v \leftarrow v * x; \\ & & w \leftarrow c * x; \\ & & y \leftarrow u + v; \\ & & y \leftarrow y + w; \\ & & y \leftarrow y + d; \\ OV & : & y. \end{array}$$

(DAG-optimal)

$$\pi' = (IV, \beta', OV) \in SLC$$

$$\begin{array}{rcl} IV & : & x; \\ \beta' & : & y \leftarrow x * a; \\ & & y \leftarrow y + b; \\ & & y \leftarrow y * x; \\ & & y \leftarrow y + c; \\ & & y \leftarrow y * x; \\ & & y \leftarrow y + d; \\ OV & : & y. \end{array}$$

Horner-Regel
$\rightsquigarrow$ Algorithmik