Compiler Construction: Assignment 1

Fabian Krause

November 8, 2021

Meta

- Name: Fabian Krause
- Contact: fabian.krause@students.uni-freiburg.de (also on course website)

Assignment 1: Partial evaluator for $\mathcal{L}_{Int}$

- Take commutativity, associativity & *negation* into account when partially evaluating arithmetic expressions
- E.g.
  - `32 + input_int() + 10` $\rightsquigarrow$ `42 + input_int()`
  - `-(-3) + (-(input_int() + (-39)))` $\rightsquigarrow$ `42 + (- input_int())`

Approach

Adjust `pe_exp` to return a *residual*:

$$\begin{aligned} \mathit{inert} &::= \texttt{input\_int}() \mid -\texttt{input\_int}() \mid \mathit{inert} + \mathit{inert} \\ \mathit{residual} &::= \mathit{int} \mid \mathit{int} + \mathit{inert} \mid \mathit{inert} \end{aligned}$$

Forces `int`'s to the left!
Change `pe_add` and `pe_neg` to take and return *residuals*:

$$\begin{aligned} \texttt{pe\_add} &: \mathit{Residual} \to \mathit{Residual} \to \mathit{Residual} \\ \texttt{pe\_neg} &: \mathit{Residual} \to \mathit{Residual} \end{aligned}$$

Change `pe_exp`:

$$\texttt{pe\_exp} : \mathit{Expression} \to \mathit{Residual}$$

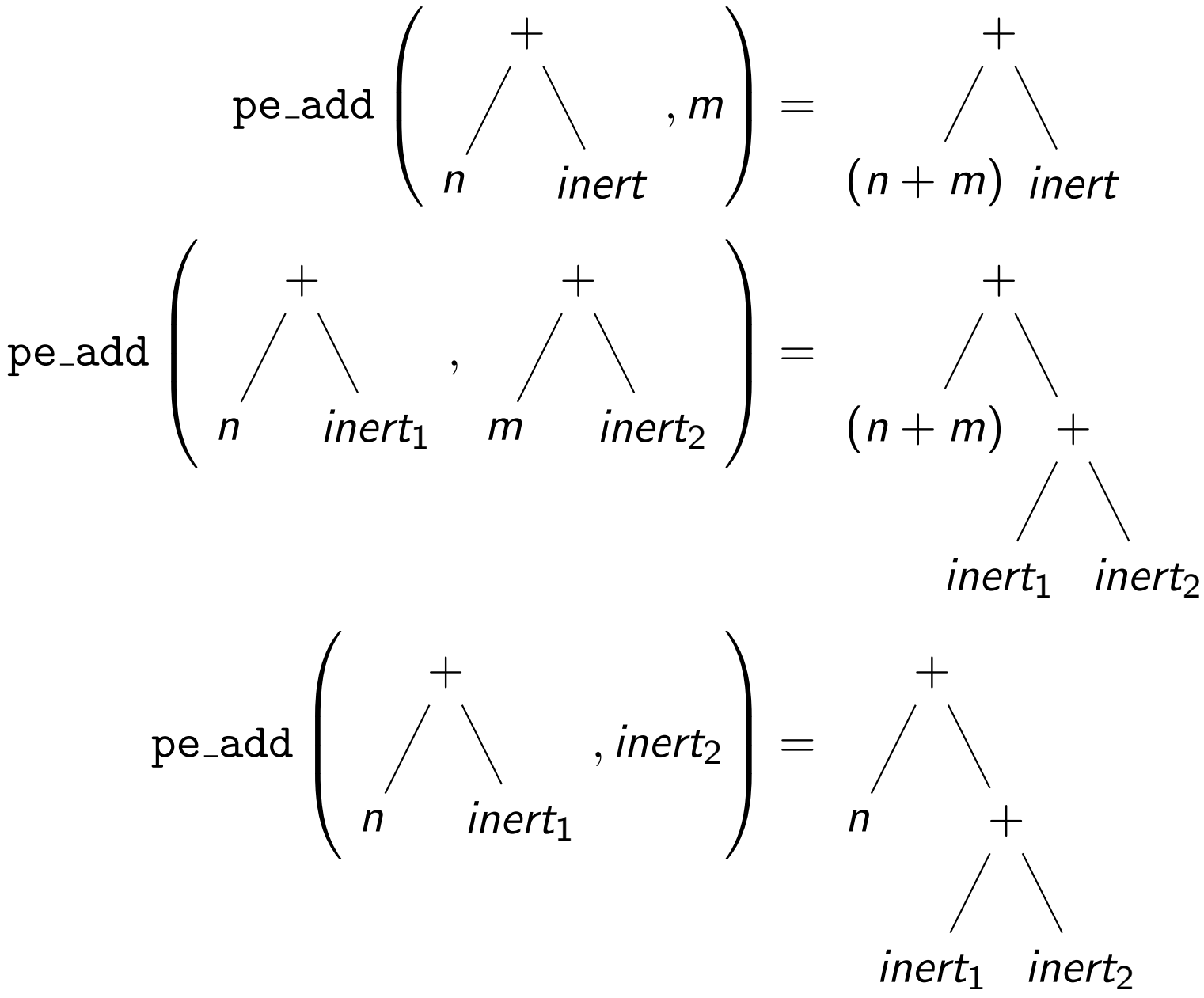
pe_add

$$\texttt{pe\_add}\,(n, m) = n + m$$

$$\texttt{pe\_add}\left(n,\ \begin{array}{ccc} & + & \\ \diagup & & \diagdown \\ m & & \mathit{inert} \end{array}\right) = \begin{array}{ccc} & + & \\ \diagup & & \diagdown \\ (n+m) & & \mathit{inert} \end{array}$$

$$\texttt{pe\_add}\,(n, \mathit{inert}) = \begin{array}{ccc} & + & \\ \diagup & & \diagdown \\ n & & \mathit{inert} \end{array}$$

pe_add cont.


pe_add
+
n
inert
, m
=
+
(n + m)
inert
pe_add
+
n
inert1
,
+
m
inert2
=
+
(n + m)
+
inert1
inert2
pe_add
+
n
inert1
, inert2
=
+
n
+
inert1
inert2

pe_add cont.

$$\texttt{pe\_add}\,(\mathit{inert}, m) = \quad \begin{array}{ccc} & + & \\ \swarrow & & \searrow \\ m & & \mathit{inert} \end{array}$$

$$\texttt{pe\_add}\left(\mathit{inert}_1, \begin{array}{ccc} & + & \\ \swarrow & & \searrow \\ m & & \mathit{inert}_2 \end{array}\right) = \begin{array}{ccccc} & & + & & \\ & \swarrow & & \searrow & \\ m & & & & + \\ & & & \swarrow & & \searrow \\ & & \mathit{inert}_1 & & & \mathit{inert}_2 \end{array}$$

$$\texttt{pe\_add}\,(\mathit{inert}_1, \mathit{inert}_2) = \quad \begin{array}{ccc} & + & \\ \swarrow & & \searrow \\ \mathit{inert}_1 & & \mathit{inert}_2 \end{array}$$

pe_neg

$$\texttt{pe\_neg}\,(n) = -n$$

$$\texttt{pe\_neg}\left(\begin{array}{ccc} & + & \\ \diagup & & \diagdown \\ n & & \mathit{inert} \end{array}\right) = \begin{array}{ccc} & + & \\ \diagup & & \diagdown \\ -n & & \texttt{pe\_neg}(\mathit{inert}) \end{array}$$

$$\texttt{pe\_neg}\,(\mathit{input\_int}()) = -\mathit{input\_int}()$$

$$\texttt{pe\_neg}\,(-\mathit{input\_int}()) = \mathit{input\_int}()$$

$$\texttt{pe\_neg}\left(\begin{array}{ccc} & + & \\ \diagup & & \diagdown \\ \mathit{inert}_1 & & \mathit{inert}_2 \end{array}\right) = \begin{array}{ccc} & + & \\ \diagup & & \diagdown \\ \texttt{pe\_neg}(\mathit{inert}_1) & & \texttt{pe\_neg}(\mathit{inert}_2) \end{array}$$

pe_exp

$$\texttt{pe\_exp}\left(\begin{array}{ccc} & + & \\ \diagup & & \diagdown \\ exp_1 & & exp_2 \end{array}\right) = \texttt{pe\_add}\left(\texttt{pe\_exp}\left(exp_1\right), \texttt{pe\_exp}\left(exp_2\right)\right)$$

$$\texttt{pe\_exp}\left(-exp\right) = \texttt{pe\_neg}\left(\texttt{pe\_exp}\left(exp\right)\right)$$

$$\texttt{pe\_exp}\left(n\right) = n$$

$$\texttt{pe\_exp}\left(input\_int()\right) = input\_int()$$

Optional Exercise

Partial evaluator for $\mathcal{L}_{Var}$

- Add an environment that maps variables to (partially) evaluated expressions
- Watch out for `input_int()` calls! Can't simply replace variables with their assigned expressions.
  - Fix: Predicate to check whether expressions includes `input_int()`
- For associativity, ...: Minor adjustments to `pe_add` and `pe_neg`, depending on the implementation.

Questions?