Piece-wise Polynomial Size Analysis for Functional Programs

O. Shkaravska M. van Eekelen A. Tamalet

Digital Security, ICIS
Radboud Universiteit Nijmegen

Seminar IOC, Tallinn, 14 Aug 2008

Sponsored by the Netherlands Organisation for Scientific Research (NWO), project Amortized Heap Space Usage Analysis (AHA), grantnr. 612.063.511.

Outline

Outline

Predict memory and time behavior

- Prevent abrupt termination: for small devices (mobile phones, Java cards, etc.), for time and memory exhaustive computations (GRID, model-generation).
- Optimize memory management (less fragmentation etc.).
- Avoid “Denial of Service” attacks that exploit memory exhaustion.
- Use in heap/stack and time properties verification.

Outline

Strict non-monotone polynomial size dependencies

Our previous work: strict polynomial size dependencies:

- size-annotated type system to check and infer f.o. types
  like: append : $\mathsf{L}_n(\alpha) \times \mathsf{L}_m(\alpha) \rightarrow \mathsf{L}_{n+m}(\alpha)$
  sqdiff : $\mathsf{L}_n(\alpha) \times \mathsf{L}_m(\alpha) \rightarrow \mathsf{L}_{(n-m)^2}(\alpha)$
- checking is decidable in integers under a syntactic restriction,
- test-based inference is semi-decidable (given a degree of polynomials, hypothetical size annotations for a f.o. type are generated via testing and checked by a type-checker).

Strict non-monotone polynomial size dependencies

Main disadvantage: cannot analyse non-strict size dependencies:

$$\text{insert}' : \text{Int} \times \text{L}_n(\text{Int}) \rightarrow \text{L}_{n,\, n+1}(\text{Int})$$
$$\text{delete}' : \text{Int} \times \text{L}_n(\text{Int}) \rightarrow \text{L}_{n,\, n\dot{-}1}(\text{Int})$$

Other work (all non-strict size dependencies):

- checking/inference is decidable in integers, but for linear polynomials (L. Pareto),
- checking/inference is decidable for monotone s.d., in reals (polynomial quasi-interpretations of J.-Y. Marion),
- decidability depends on external packages (K. Hammond),
- monotone, inference with some human interaction (G. Puebla),
- size dependencies as programs (B. Jay)

Strict non-monotone polynomial size dependencies

Main disadvantage: cannot analyse non-strict size dependencies:

$$\text{insert}' : \text{Int} \times \text{L}_n(\text{Int}) \rightarrow \text{L}_{n,\, n+1}(\text{Int})$$
$$\text{delete}' : \text{Int} \times \text{L}_n(\text{Int}) \rightarrow \text{L}_{n,\, n\dot{-}1}(\text{Int})$$

Other work (all non-strict size dependencies):

- checking/inference is decidable in integers, but for linear polynomials (L. Pareto),
- checking/inference is decidable for monotone s.d., in reals (polynomial quasi-interpretations of J.-Y. Marion),
- decidability depends on external packages (K. Hammond),
- monotone, inference with some human interaction (G. Puebla),
- size dependencies as programs (B. Jay)

Outline

The language

$$
\begin{array}{llll}
\textit{Basic} & b & ::= & c \mid \mathrm{Nil} \mid \mathrm{Cons}(\mathrm{x}, \mathrm{y}) \mid f(\mathrm{x}_1, \ldots, \mathrm{x}_n) \\
\textit{Expr} & e & ::= & \mathrm{letfun}\ f(\mathrm{x}_1, \ldots, \mathrm{x}_n) = e_1\ \mathrm{in}\ e_2 \\
 & & & \mid b \\
 & & & \mid \mathrm{let}\ \mathrm{x} = b\ \mathrm{in}\ e \\
 & & & \mid \mathrm{if}\ \mathrm{x}\ \mathrm{then}\ e_1\ \mathrm{else}\ e_2 \\
 & & & \mid \mathrm{match}\ \mathrm{x}\ \mathrm{with} \mid \mathrm{Nil} \Rightarrow e_1 \\
 & & & \qquad\qquad \mid \mathrm{Cons}(\mathrm{hd}, \mathrm{tl}) \Rightarrow e_2
\end{array}
$$

Zero-order types

Zero-order types	are (still) matrix-like structures:
$[1, 2]$	$\in L_2(\mathrm{Int})$
$[[1, 3], [1, 4], [1, 5]]$	$\in L_3(L_2(\mathrm{Int}))$

... and unions of them

$[1, 2]$	$\in L_1(\mathrm{Int}) \cup L_2(\mathrm{Int})$
	$= L_i^{1 \le i \le 2}(\mathrm{Int})$
$[[1, 3], [1, 4], [1, 5]]$	$\in L_3(L_1(\mathrm{Int})) \cup L_3(L_2(\mathrm{Int}))$
formally	$= \exists\, 1 \le i \le 2.\, L_3(L_i(\mathrm{Int}))$
notation	$= L_3(L_i^{1 \le i \le 2}(\mathrm{Int}))$

Zero-order types

Zero-order types	are (still) matrix-like structures:
$[1, 2]$	$\in L_2(\mathrm{Int})$
$[[1, 3], [1, 4], [1, 5]]$	$\in L_3(L_2(\mathrm{Int}))$

... and unions of them

$[1, 2]$	$\in L_1(\mathrm{Int}) \cup L_2(\mathrm{Int})$
	$= L_i^{1 \le i \le 2}(\mathrm{Int})$
$[[1, 3], [1, 4], [1, 5]]$	$\in L_3(L_1(\mathrm{Int})) \cup L_3(L_2(\mathrm{Int}))$
formally	$= \exists\, 1 \le i \le 2.\, L_3(L_i(\mathrm{Int}))$
notation	$= L_3(L_i^{1 \le i \le 2}(\mathrm{Int}))$

Zero-order types

An example of types	with size/type variables:
$\mathsf{L}_{n}(\alpha)$	(*formal-parameter types for functions*)
$\mathsf{L}_{n^2+i}^{0\leq i\leq n}(\alpha)$	(*output types for functions*)

$$\textit{Types}\quad \tau \quad ::= \quad \mathsf{Int} \mid \mathsf{Bool} \mid \alpha \mid \mathsf{L}_{p(\overline{n},\overline{i})}^{P(\overline{n},\overline{i})}(\tau),$$

where $P(\overline{n}, \overline{i})$ is an arithmetic quantifier-free predicate,
$p(\overline{n}, \overline{i})$ is a piece-wise polynomial (with both, $-$ and $\dot{-}$)

Semantics in an example:

$$x : \mathsf{L}_{m\dot{-}j}^{0\leq j\leq n}(\mathsf{L}_{mn+i}^{0\leq i\leq n^2}(\alpha)) \;\Big|\; \begin{array}{l}\exists\ 0 \leq j \leq n,\ 0 \leq i \leq n^2 \\ x : \mathsf{L}_{m\dot{-}j}(\mathsf{L}_{mn+i}(\alpha))\end{array}$$

Zero-order types

An example of types	with size/type variables:
$\mathsf{L}_{n}(\alpha)$	(*formal-parameter types for functions*)
$\mathsf{L}_{n^2+i}^{0\leq i\leq n}(\alpha)$	(*output types for functions*)

$$\textit{Types}\quad \tau \quad ::= \quad \mathsf{Int} \mid \mathsf{Bool} \mid \alpha \mid \mathsf{L}_{p(\overline{n},\overline{i})}^{P(\overline{n},\overline{i})}(\tau),$$

where $P(\overline{n}, \overline{i})$ is an arithmetic quantifier-free predicate, $p(\overline{n}, \overline{i})$ is a piece-wise polynomial (with both, $-$ and $\dot{-}$)

Semantics in an example:

$$x : \mathsf{L}_{m\dot{-}j}^{0\leq j\leq n}(\mathsf{L}_{mn+i}^{0\leq i\leq n^2}(\alpha)) \;\Big|\; \begin{array}{l} \exists\ 0 \leq j \leq n,\ 0 \leq i \leq n^2 \\ x : \mathsf{L}_{m\dot{-}j}(\mathsf{L}_{mn+i}(\alpha)) \end{array}$$

Function types

$$
\begin{array}{lll}
\text{insert} & : (\alpha \times \alpha \to \text{Bool}) \times \alpha \times \mathsf{L}_n(\alpha) & \to \mathsf{L}_{n+i}^{0 \le i \le 1}(\alpha) \\
\text{rinsert} & : (\alpha \times \alpha \to \text{Bool}) \times \mathsf{L}_n(\alpha) \times \mathsf{L}_m(\alpha) & \to \mathsf{L}_{m+i}^{0 \le i \le n}(\alpha) \\
\text{filter} & : (\alpha \times \alpha \to \text{Bool}) \times \mathsf{L}_n(\alpha) & \to \mathsf{L}_{i}^{0 \le i \le n}(\alpha) \\
\text{delete} & : (\alpha \times \alpha \to \text{Bool}) \times \alpha \times \mathsf{L}_n(\alpha) & \to \mathsf{L}_{n \dot{-} i}^{0 \le i \le 1}(\alpha) \\
\text{rdelete} & : (\alpha \times \alpha \to \text{Bool}) \times \mathsf{L}_n(\alpha) \times \mathsf{L}_m(\alpha) & \to \mathsf{L}_{m \dot{-} i}^{0 \le i \le n}(\alpha) \\
\text{divtwo} & : \mathsf{L}_n(\alpha) & \to \mathsf{L}_{\frac{n \dot{-} i}{2}}^{0 \le i \le 1}(\alpha)
\end{array}
$$

- formal-parameter types: zero-order with just-size-variable-annotations, and higher-order.
- Output types: zero-order annotations that do not depend on the annotations (if any) of the higher-order arguments.

Function types

$$
\begin{array}{lll}
\text{insert} & : (\alpha \times \alpha \to \text{Bool}) \times \alpha \times \mathsf{L}_n(\alpha) & \to \mathsf{L}_{n+i}^{0 \le i \le 1}(\alpha) \\
\text{rinsert} & : (\alpha \times \alpha \to \text{Bool}) \times \mathsf{L}_n(\alpha) \times \mathsf{L}_m(\alpha) & \to \mathsf{L}_{m+i}^{0 \le i \le n}(\alpha) \\
\text{filter} & : (\alpha \times \alpha \to \text{Bool}) \times \mathsf{L}_n(\alpha) & \to \mathsf{L}_{i}^{0 \le i \le n}(\alpha) \\
\text{delete} & : (\alpha \times \alpha \to \text{Bool}) \times \alpha \times \mathsf{L}_n(\alpha) & \to \mathsf{L}_{n \dot{-} i}^{0 \le i \le 1}(\alpha) \\
\text{rdelete} & : (\alpha \times \alpha \to \text{Bool}) \times \mathsf{L}_n(\alpha) \times \mathsf{L}_m(\alpha) & \to \mathsf{L}_{m \dot{-} i}^{0 \le i \le n}(\alpha) \\
\text{divtwo} & : \mathsf{L}_n(\alpha) & \to \mathsf{L}_{\frac{n \dot{-} i}{2}}^{0 \le i \le 1}(\alpha)
\end{array}
$$

- formal-parameter types: zero-order with just-size-variable-annotations, and higher-order.
- Output types: zero-order annotations that do not depend on the annotations (if any) of the higher-order arguments.

Function types

$$
\begin{array}{lll}
\text{insert} & : (\alpha \times \alpha \to \text{Bool}) \times \alpha \times \mathsf{L}_n(\alpha) & \to \mathsf{L}_{n+i}^{0 \le i \le 1}(\alpha) \\
\text{rinsert} & : (\alpha \times \alpha \to \text{Bool}) \times \mathsf{L}_n(\alpha) \times \mathsf{L}_m(\alpha) & \to \mathsf{L}_{m+i}^{0 \le i \le n}(\alpha) \\
\text{filter} & : (\alpha \times \alpha \to \text{Bool}) \times \mathsf{L}_n(\alpha) & \to \mathsf{L}_{i}^{0 \le i \le n}(\alpha) \\
\text{delete} & : (\alpha \times \alpha \to \text{Bool}) \times \alpha \times \mathsf{L}_n(\alpha) & \to \mathsf{L}_{n \dot{-} i}^{0 \le i \le 1}(\alpha) \\
\text{rdelete} & : (\alpha \times \alpha \to \text{Bool}) \times \mathsf{L}_n(\alpha) \times \mathsf{L}_m(\alpha) & \to \mathsf{L}_{m \dot{-} i}^{0 \le i \le n}(\alpha) \\
\text{divtwo} & : \mathsf{L}_n(\alpha) & \to \mathsf{L}_{\frac{n \dot{-} i}{2}}^{0 \le i \le 1}(\alpha)
\end{array}
$$

- formal-parameter types: zero-order with just-size-variable-annotations, and higher-order.
- Output types: zero-order annotations that do not depend on the annotations (if any) of the higher-order arguments.

Function types

$$
\begin{array}{lcll}
\text{insert} & : & (\alpha \times \alpha \to \text{Bool}) \times \alpha \times \mathsf{L}_n(\alpha) & \to \quad \mathsf{L}_{n+i}^{0 \le i \le 1}(\alpha) \\
\text{rinsert} & : & (\alpha \times \alpha \to \text{Bool}) \times \mathsf{L}_n(\alpha) \times \mathsf{L}_m(\alpha) & \to \quad \mathsf{L}_{m+i}^{0 \le i \le n}(\alpha) \\
\text{filter} & : & (\alpha \times \alpha \to \text{Bool}) \times \mathsf{L}_n(\alpha) & \to \quad \mathsf{L}_{i}^{0 \le i \le n}(\alpha) \\
\text{delete} & : & (\alpha \times \alpha \to \text{Bool}) \times \alpha \times \mathsf{L}_n(\alpha) & \to \quad \mathsf{L}_{n \dot{-} i}^{0 \le i \le 1}(\alpha) \\
\text{rdelete} & : & (\alpha \times \alpha \to \text{Bool}) \times \mathsf{L}_n(\alpha) \times \mathsf{L}_m(\alpha) & \to \quad \mathsf{L}_{m \dot{-} i}^{0 \le i \le n}(\alpha) \\
\text{divtwo} & : & \mathsf{L}_n(\alpha) & \to \quad \mathsf{L}_{\frac{n \dot{-} i}{2}}^{0 \le i \le 1}(\alpha)
\end{array}
$$

- formal-parameter types: zero-order with just-size-variable-annotations, and higher-order.
- Output types: zero-order annotations that do not depend on the annotations (if any) of the higher-order arguments.

Function types

$$
\begin{array}{lcll}
\text{insert} & : & (\alpha \times \alpha \to \text{Bool}) \times \alpha \times \mathsf{L}_n(\alpha) & \to \quad \mathsf{L}_{n+i}^{0 \le i \le 1}(\alpha) \\
\text{rinsert} & : & (\alpha \times \alpha \to \text{Bool}) \times \mathsf{L}_n(\alpha) \times \mathsf{L}_m(\alpha) & \to \quad \mathsf{L}_{m+i}^{0 \le i \le n}(\alpha) \\
\text{filter} & : & (\alpha \times \alpha \to \text{Bool}) \times \mathsf{L}_n(\alpha) & \to \quad \mathsf{L}_{i}^{0 \le i \le n}(\alpha) \\
\text{delete} & : & (\alpha \times \alpha \to \text{Bool}) \times \alpha \times \mathsf{L}_n(\alpha) & \to \quad \mathsf{L}_{n \dot{-} i}^{0 \le i \le 1}(\alpha) \\
\text{rdelete} & : & (\alpha \times \alpha \to \text{Bool}) \times \mathsf{L}_n(\alpha) \times \mathsf{L}_m(\alpha) & \to \quad \mathsf{L}_{m \dot{-} i}^{0 \le i \le n}(\alpha) \\
\text{divtwo} & : & \mathsf{L}_n(\alpha) & \to \quad \mathsf{L}_{\frac{n \dot{-} i}{2}}^{0 \le i \le 1}(\alpha)
\end{array}
$$

- formal-parameter types: zero-order with just-size-variable-annotations, and higher-order.
- Output types: zero-order annotations that do not depend on the annotations (if any) of the higher-order arguments.

Typing rules

$$\frac{D \vdash p(\overline{n}) = p'(\overline{n}) + 1}{D;\ \Gamma,\ \mathsf{hd} : \tau,\ \mathsf{tl} : \mathsf{L}_{p'(\overline{n})}(\tau) \vdash_\Sigma \mathsf{Cons}(\mathsf{hd}, \mathsf{tl}) : \mathsf{L}_{p(\overline{n})}(\tau)}\ \textsc{Cons} - \textit{old}$$

$$\frac{D(\overline{n}, \overline{j}) \vdash \mathsf{L}^{Q(\overline{n}, \overline{i})}_{p(\overline{n}, \overline{i})}(\tau) \triangleleft \mathsf{L}^{Q'(\overline{n}, \overline{j'})}_{p'(\overline{n}, \overline{j'})+1}(\tau')}{\begin{array}{l} D(\overline{n}, \overline{j});\ \Gamma,\ \mathsf{hd} : \tau',\ \mathsf{tl} : \mathsf{L}^{Q'(\overline{n}, \overline{j'})}_{p'(\overline{n}, \overline{j'})}(\tau') \vdash_\Sigma \\ \mathsf{Cons}(\mathsf{hd}, \mathsf{tl}) :\ \mathsf{L}^{Q(\overline{n}, \overline{i})}_{p(\overline{n}, \overline{i})}(\tau) \end{array}}\ \textsc{Cons}$$

$$D(\overline{n}, \overline{j}) \vdash \mathsf{L}^{Q(\overline{n}, \overline{i})}_{p(\overline{n}, \overline{i})}(\tau) \triangleleft \mathsf{L}^{Q'(\overline{n}, \overline{j'})}_{p'(\overline{n}, \overline{j'})+1}(\tau') \quad \Big| \quad \forall\, \overline{n}\, \overline{j}\, \overline{j'}\ \exists \overline{i}.\ D(\overline{n}, \overline{j}) \wedge Q'(\overline{n}, \overline{j'}) \Rightarrow Q(\overline{n}, \overline{i}) \wedge p(\overline{n}, \overline{i}) = p'(\overline{n}, \overline{j'}) + 1$$

Typing rules

$$\frac{D \vdash p(\overline{n}) = p'(\overline{n}) + 1}{D;\ \Gamma,\ \mathsf{hd}:\tau,\ \mathsf{tl}:\mathsf{L}_{p'(\overline{n})}(\tau) \vdash_\Sigma \mathsf{Cons}(\mathsf{hd},\mathsf{tl}):\mathsf{L}_{p(\overline{n})}(\tau)}\ \textsc{Cons}-\textit{old}$$

$$\frac{D(\overline{n},\overline{j}) \vdash \mathsf{L}^{Q(\overline{n},\overline{i})}_{p(\overline{n},\overline{i})}(\tau) \lhd \mathsf{L}^{Q'(\overline{n},\overline{j}')}_{p'(\overline{n},\overline{j}')+1}(\tau')}{\begin{array}{l} D(\overline{n},\overline{j});\ \Gamma,\ \mathsf{hd}:\tau',\ \mathsf{tl}:\mathsf{L}^{Q'(\overline{n},\overline{j}')}_{p'(\overline{n},\overline{j}')}(\tau') \vdash_\Sigma \\ \mathsf{Cons}(\mathsf{hd},\mathsf{tl}):\ \mathsf{L}^{Q(\overline{n},\overline{i})}_{p(\overline{n},\overline{i})}(\tau) \end{array}}\ \textsc{Cons}$$

$$D(\overline{n},\overline{j}) \vdash \mathsf{L}^{Q(\overline{n},\overline{i})}_{p(\overline{n},\overline{i})}(\tau) \lhd \mathsf{L}^{Q'(\overline{n},\overline{j}')}_{p'(\overline{n},\overline{j}')+1}(\tau') \quad \Big| \quad \forall\, \overline{n}\,\overline{j}\,\overline{j}'\ \exists \overline{i}.\ D(\overline{n},\overline{j}) \wedge Q'(\overline{n},\overline{j}') \Rightarrow Q(\overline{n},\overline{i}) \wedge p(\overline{n},\overline{i}) = p'(\overline{n},\overline{j}') + 1$$

Typing rules

$$\frac{D \vdash p(\overline{n}) = p'(\overline{n}) + 1}{D;\ \Gamma,\ \mathsf{hd}:\tau,\ \mathsf{tl}:\mathsf{L}_{p'(\overline{n})}(\tau) \vdash_\Sigma \mathsf{Cons}(\mathsf{hd},\mathsf{tl}):\mathsf{L}_{p(\overline{n})}(\tau)}\ \textsc{Cons}-\textit{old}$$

$$\frac{D(\overline{n},\overline{j}) \vdash \mathsf{L}^{Q(\overline{n},\overline{i})}_{p(\overline{n},\overline{i})}(\tau) \lhd \mathsf{L}^{Q'(\overline{n},\overline{j}')}_{p'(\overline{n},\overline{j}')+1}(\tau')}{\begin{array}{l} D(\overline{n},\overline{j});\ \Gamma,\ \mathsf{hd}:\tau',\ \mathsf{tl}:\mathsf{L}^{Q'(\overline{n},\overline{j}')}_{p'(\overline{n},\overline{j}')}(\tau') \vdash_\Sigma \\ \mathsf{Cons}(\mathsf{hd},\mathsf{tl}):\ \mathsf{L}^{Q(\overline{n},\overline{i})}_{p(\overline{n},\overline{i})}(\tau) \end{array}}\ \textsc{Cons}$$

$$D(\overline{n},\overline{j}) \vdash \mathsf{L}^{Q(\overline{n},\overline{i})}_{p(\overline{n},\overline{i})}(\tau) \lhd \mathsf{L}^{Q'(\overline{n},\overline{j}')}_{p'(\overline{n},\overline{j}')+1}(\tau') \quad \Big| \quad \forall\, \overline{n}\,\overline{j}\,\overline{j}'\ \exists \overline{i}.\ D(\overline{n},\overline{j}) \wedge Q'(\overline{n},\overline{j}') \Rightarrow Q(\overline{n},\overline{i}) \wedge p(\overline{n},\overline{i}) = p'(\overline{n},\overline{j}') + 1$$

Typing rules

$$\frac{\Sigma(f) = \tau_1^f \times \ldots \times \tau_{k'}^f \times \tau_1^\circ \times \ldots \times \tau_k^\circ \to \tau_0 \quad \Sigma(g_1) = \tau_1^f, \ldots, \Sigma(g_{k'}) = \tau_{k'}^f \quad D \vdash \tau_0' \lhd *(\tau_0)}{D;\ \Gamma, x_1 : \tau_1', \ldots, x_1 : \tau_k' \vdash_\Sigma f(g_1, \ldots, g_{k'},\ x_1, \ldots, x_k) : \tau_0'}\ \textsc{FApp}$$

Substitution $*$ on free size parameters

input $\quad \mathsf{L}_m(-) \quad \mapsto \quad \mathsf{L}^{P(\overline{n},\overline{i})}_{p(\overline{n},\overline{i})}(-)$

output $\quad \mathsf{L}^{Q(\ldots,m,\ldots,\overline{j})}_{q(\ldots,m,\ldots,\overline{j})}(\ \ ) \mapsto \quad \mathsf{L}^{P(\overline{n},\overline{i}) \wedge Q(\ldots,p(\overline{n},\overline{i}),\ldots\overline{j})}_{q(\ldots,p(\overline{n},\overline{i}),\ldots,\overline{j})}(-)$

"Collections-of-polynomials" annotations handle non-monotone bounds.

Typing rules

$$\frac{\Sigma(f) = \tau_1^f \times \ldots \times \tau_{k'}^f \times \tau_1^\circ \times \ldots \times \tau_k^\circ \to \tau_0 \quad \Sigma(g_1) = \tau_1^f, \ldots, \Sigma(g_{k'}) = \tau_{k'}^f \quad D \vdash \tau_0' \lhd *(\tau_0)}{D;\ \Gamma, x_1 : \tau_1', \ldots, x_1 : \tau_k' \vdash_\Sigma f(g_1, \ldots, g_{k'},\ x_1, \ldots, x_k) : \tau_0'}\ \textsc{FApp}$$

Substitution $*$ on free size parameters

$$\text{input} \quad \mathsf{L}_m(-) \mapsto \mathsf{L}^{P(\overline{n},\overline{i})}_{p(\overline{n},\overline{i})}(-)$$

$$\text{output} \quad \mathsf{L}^{Q(\ldots,m,\ldots,\overline{j})}_{q(\ldots,m,\ldots,\overline{j})}(\ \ ) \mapsto \mathsf{L}^{P(\overline{n},\overline{i}) \wedge Q(\ldots,p(\overline{n},\overline{i}),\ldots\overline{j})}_{q(\ldots,p(\overline{n},\overline{i}),\ldots,\overline{j})}(-)$$

"Collections-of-polynomials" annotations handle non-monotone bounds.

Typing rules

The IF-rule: the same types in both branches, but its existentials may be instantiated with different values of $\bar{i}$:

$$\frac{\Gamma(x) = \mathsf{Bool} \quad D;\ \Gamma \vdash_\Sigma e_t : \tau \quad D;\ \Gamma \vdash_\Sigma e_f : \tau}{D;\ \Gamma \vdash_\Sigma \text{if } x \text{ then } e_t \text{ else } e_f : \tau}\ \textsc{If}$$

Outline

Type checking: example

$\text{insert} : (\alpha \times \alpha \to \text{Bool}) \times \alpha \times \text{L}_n(\alpha) \to \text{L}_{n+i}^{0 \le i \le 1}(\alpha)$:

$$
\begin{array}{l}
\text{insert(g, x, y)} = \\
\text{match y with} \mid \text{Nil} \Rightarrow \text{let z} = \text{Nil in Cons(x, z)} \\
\qquad\qquad\qquad\quad \mid \text{Cons(hd, tl)} \Rightarrow \quad \text{if g(x, hd)} \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad \text{then y} \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad \text{else Cons(hd, insert(g, x, tl))}
\end{array}
$$

$$
\begin{array}{lcl}
n = 0 & \vdash & n{+}?i = 0 \wedge 0 \le ?i \le 1 \\
n > 0 & \vdash & n{+}?i = n \wedge 0 \le ?i \le 1 \\
n > 0;\ 0 \le j \le 1 & \vdash & n{+}?i = (n-1) + j + 1 \wedge 0 \le ?i \le 1
\end{array}
$$

Type checking: example

$$
\begin{array}{lcl}
n = 0 & \vdash & n{+}?i = 0 \wedge 0 \leq ?i \leq 1 \\
n > 0 & \vdash & n{+}?i = n \wedge 0 \leq ?i \leq 1 \\
n > 0;\ 0 \leq j \leq 1 & \vdash & n{+}?i = (n-1) + j + 1 \wedge 0 \leq ?i \leq 1
\end{array}
$$

Solution:

$$
\begin{array}{lcl}
 & \vdash & ?i := 0 \wedge 0 \leq ?i \leq 1 \\
n > 0 & \vdash & ?i := n - n = 0 \wedge 0 \leq ?i \leq 1 \\
n > 0;\ 0 \leq j \leq 1 & \vdash & ?i := j \wedge 0 \leq ?i \leq 1
\end{array}
$$

Output size annotations

$$\begin{array}{lcl} \text{rinsert} & : & (\alpha \times \alpha \to \text{Bool}) \times \text{L}_n(\alpha) \times \text{L}_m(\alpha) \to \text{L}_{m+i}^{0 \le i \le n}(\alpha) \\ \text{rdelete} & : & (\alpha \times \alpha \to \text{Bool}) \times \text{L}_n(\alpha) \times \text{L}_m(\alpha) \to \text{L}_{m \dot{-} i}^{0 \le i \le n}(\alpha) \end{array}$$

Size annot. of the form	$p(\overline{n}) + i$ or $p(\overline{n}) \dot{-} i$
$Q(\overline{n}, i)$ of the form	$0 \le i \le \delta(\overline{n})$

This

1. makes type checking easier, since end entailments are of the form
   - $D(\overline{n}, \overline{j}) \vdash p(\overline{n}) + i = q(\overline{n}, \overline{j}) \wedge Q(\overline{n}, i)$
     i.e. to check $D(\overline{n}, \overline{j}) \vdash Q(\overline{n}, q(\overline{n}, \overline{j}) - p(\overline{n}))$,
   - or $D(\overline{n}, \overline{j}) \vdash p(\overline{n}) \dot{-} i = q(\overline{n}, \overline{j}) \wedge Q(\overline{n}, i)$
     i.e. to check ... see the next slide ...
2. comes from one natural observation (see the next slides).

Output size annotations

$$\begin{array}{lcl} \text{rinsert} & : & (\alpha \times \alpha \to \text{Bool}) \times \text{L}_n(\alpha) \times \text{L}_m(\alpha) \to \text{L}_{m+i}^{0 \le i \le n}(\alpha) \\ \text{rdelete} & : & (\alpha \times \alpha \to \text{Bool}) \times \text{L}_n(\alpha) \times \text{L}_m(\alpha) \to \text{L}_{m \dot{-} i}^{0 \le i \le n}(\alpha) \end{array}$$

Size annot. of the form	$p(\overline{n}) + i$ or $p(\overline{n}) \dot{-} i$
$Q(\overline{n}, i)$ of the form	$0 \le i \le \delta(\overline{n})$

This

1. makes type checking easier, since end entailments are of the form
   - $D(\overline{n}, \overline{j}) \vdash p(\overline{n}) + i = q(\overline{n}, \overline{j}) \wedge Q(\overline{n}, i)$
     i.e. to check $D(\overline{n}, \overline{j}) \vdash Q(\overline{n}, q(\overline{n}, \overline{j}) - p(\overline{n}))$,
   - or $D(\overline{n}, \overline{j}) \vdash p(\overline{n}) \dot{-} i = q(\overline{n}, \overline{j}) \wedge Q(\overline{n}, i)$
     i.e. to check ... see the next slide ...
2. comes from one natural observation (see the next slides).

Output size annotations

Size annot. of the form	$p(\overline{n}) + i$ or $p(\overline{n}) \dot{-} i$
$Q(\overline{n},\ i)$ of the form	$0 \leq i \leq \delta(\overline{n})$

This

1. makes type checking easier, since end entailments are of the form
   - (see above),
   - or $D(\overline{n}, \overline{j}) \vdash\ p(\overline{n}) \dot{-} i = q(\overline{n},\ \overline{j}) \wedge Q(\overline{n},\ i)$

     i.e. to check one of the

     $$D(\overline{n}, \overline{j}),\ p(\overline{n}) - q(\overline{n},\ \overline{j}) \leq p(\overline{n}) \quad \vdash \quad Q(\overline{n},\ p(\overline{n}) - q(\overline{n},\ \overline{j}))\ ,$$
     $$D(\overline{n}, \overline{j}),\ i > p(\overline{n}) \quad \vdash \quad q(\overline{n},\ \overline{j}) = 0 \wedge Q(\overline{n},\ i)$$
2. comes from one natural observation (see the next slide)

Output size annotations

Size annot. of the form	$p(\overline{n}) + i$ or $p(\overline{n}) \dot{-} i$
$Q(\overline{n},\ i)$ of the form	$0 \leq i \leq \delta(\overline{n})$

This

1. makes type checking easier, since end entailments are of the form
   - (see above),
   - or $D(\overline{n}, \overline{j}) \vdash\ p(\overline{n}) \dot{-} i = q(\overline{n},\ \overline{j}) \wedge Q(\overline{n},\ i)$
     i.e. to check one of the
     $$D(\overline{n}, \overline{j}),\ p(\overline{n}) - q(\overline{n},\ \overline{j}) \leq p(\overline{n}) \quad \vdash \quad Q(\overline{n},\ p(\overline{n}) - q(\overline{n},\ \overline{j}))\ ,$$
     $$D(\overline{n}, \overline{j}),\ i > p(\overline{n}) \quad \vdash \quad q(\overline{n},\ \overline{j}) = 0 \wedge Q(\overline{n},\ i)$$
2. comes from one natural observation (see the next slide)

Output size annotations

Size annot. of the form	$p(\overline{n}) + i$ or $p(\overline{n}) \dot{-} i$
$Q(\overline{n}, i)$ of the form	$0 \leq i \leq \delta(\overline{n})$

This

1. makes type checking easier,
2. comes from the fact, that one would like to check/infer a lower $p_{min}(\overline{n})$ and an upper $p_{max}(\overline{n})$ bounds.
   This means that the length of the output value is *exactly*
   either $p_{min}(\overline{n}) + i$ for some $0 \leq i \leq p_{max}(\overline{n}) - p_{min}(\overline{n})$
   or $p_{max}(\overline{n}) \dot{-} i$ for some $0 \leq i \leq p_{max}(\overline{n}) - p_{min}(\overline{n})$

Checking using reals (CAD)

Real arithmetic is inevitable.

Just embedding integers into reals is not enough!

E.g. $x^2 \leq x^3$ is "true" for integers and "false" for reals.

Use CAD (Cylindrical Algebraic Decompositions):

to solve	$D(\overline{n}, \overline{j}) \vdash Q(\overline{n}, \overline{j})$
i.e. to find an integer counterexample $(\overline{n}, \overline{j})$	$D(\overline{n}, \overline{j}) \wedge \neg Q(\overline{n}, \overline{j})$

Checking using reals (CAD)

Real arithmetic is inevitable.

Just embedding integers into reals is not enough!

E.g. $x^2 \leq x^3$ is "true" for integers and "false" for reals.

Use CAD (Cylindrical Algebraic Decompositions):

to solve	$D(\overline{n}, \overline{j}) \vdash Q(\overline{n}, \overline{j})$
i.e. to find an integer counterexample $(\overline{n}, \overline{j})$	$D(\overline{n}, \overline{j}) \wedge \neg Q(\overline{n}, \overline{j})$

Checking using reals (CAD)

CAD for a real predicate $P(\bar{x})$

$g_{11} \leq x_1 \leq g_{12}$
$g_{21}(x_1) \leq x_2 \leq g_{22}(x_1)$
...
where g_{ij} contains $+$, $-$, $*$ and radicals.

The question: are there integer numbers in the CAD for $D(\bar{n}, \bar{j}) \wedge \neg Q(\bar{n}, \bar{j})$?
In the example: $x^2 > x^3$ holds on $0 < x < 1$,
which does not contain integers.

An easy question if g_{12} is not ∞ (enumeration of integers from bounded cylinders is used in *Mathematica*). Problem: $g_{12} = \infty$.

Checking using reals (CAD)

CAD for a real predicate $P(\bar{x})$

$g_{11} \leq x_1 \leq g_{12}$
$g_{21}(x_1) \leq x_2 \leq g_{22}(x_1)$
...
where g_{ij} contains $+$, $-$, $*$ and radicals.

The question: are there integer numbers in the CAD for $D(\bar{n}, \bar{j}) \wedge \neg Q(\bar{n}, \bar{j})$?
In the example: $x^2 > x^3$ holds on $0 < x < 1$,
which does not contain integers.

An easy question if g_{12} is not ∞ (enumeration of integers from bounded cylinders is used in *Mathematica*). Problem: $g_{12} = \infty$.

Checking using reals (CAD)

CAD for a real predicate $P(\bar{x})$

$g_{11} \leq x_1 \leq g_{12}$
$g_{21}(x_1) \leq x_2 \leq g_{22}(x_1)$
...
where g_{ij} contains $+$, $-$, $*$ and radicals.

The question: are there integer numbers in the CAD for $D(\bar{n}, \bar{j}) \wedge \neg Q(\bar{n}, \bar{j})$?
In the example: $x^2 > x^3$ holds on $0 < x < 1$,
which does not contain integers.

An easy question if g_{12} is not ∞ (enumeration of integers from bounded cylinders is used in *Mathematica*). Problem: $g_{12} = \infty$.

Outline

Inference via abstract "testing"

Assumption for a function f :

for any $\overline{n}$ (except in the base of recursion) there exists
an input $\overline{x}_{min}$ s.t. $|f(\overline{x})| = p_{min}(\overline{n})$
an input $\overline{x}_{max}$ s.t. $|f(\overline{x})| = p_{max}(\overline{n})$

An example: insert with a hypothesis $p_{min}(n) = an + b$
$p_{max}(n) = a'n + b'$

Abstract interpretation for insert: $p(0) \longrightarrow 1$
$p(n) \longrightarrow n \,|\, 1 + p(n-1)$

Inference via abstract "testing"

Assumption for a function f :

for any $\overline{n}$ (except in the base of recursion) there exists
an input $\overline{x}_{min}$ s.t. $|f(\overline{x})| = p_{min}(\overline{n})$
an input $\overline{x}_{max}$ s.t. $|f(\overline{x})| = p_{max}(\overline{n})$

An example: insert with a hypothesis $p_{min}(n) = an + b$
$p_{max}(n) = a'n + b'$

Abstract interpretation for insert: $p(0) \to 1$
$p(n) \to n \,|\, 1 + p(n-1)$

Inference via abstract "testing"

Assumption for a function f :

for any $\overline{n}$ (except in the base of recursion) there exists
an input $\overline{x}_{min}$ s.t. $|f(\overline{x})| = p_{min}(\overline{n})$
an input $\overline{x}_{max}$ s.t. $|f(\overline{x})| = p_{max}(\overline{n})$

An example: insert with a hypothesis $p_{min}(n) = an + b$
$p_{max}(n) = a'n + b'$

Abstract interpretation for insert: $p(0) \longrightarrow 1$
$p(n) \longrightarrow n \,|\, 1 + p(n-1)$

Inference via abstract "testing"

Abstract interpretation for insert: $p(0) \longrightarrow 1$
$p(n) \longrightarrow n \,|\, 1 + p(n-1)$

$$\left.\begin{array}{l} p(1) = \{1, 2\} \\ p(2) = \{2, 3\} \end{array}\right\} \Rightarrow \left\{\begin{array}{l} p_{min}(1) = 1,\ p_{min}(2) = 2 \\ p_{max}(1) = 2,\ p_{max}(2) = 3 \end{array}\right.$$

Solving the system of linear equation gives

$$p_{min}(n) = n$$
$$p_{max}(n) = n + 1$$

Inference via abstract "testing"

Abstract interpretation for insert: $p(0) \rightarrow 1$
$p(n) \rightarrow n \,|\, 1 + p(n-1)$

$$\left.\begin{array}{l} p(1) = \{1, 2\} \\ p(2) = \{2, 3\} \end{array}\right\} \Rightarrow \left\{\begin{array}{l} p_{min}(1) = 1,\ p_{min}(2) = 2 \\ p_{max}(1) = 2,\ p_{max}(2) = 3 \end{array}\right.$$

Solving the system of linear equation gives

$$p_{min}(n) = n$$
$$p_{max}(n) = n + 1$$

Summary

- a polynomial-size-annotated type system is designed,
- checking is decidable in reals and is, basically, adjusted for integers,
- test-based inference of polynomial lower and upper bounds is possible.

Future work:

- test-based inference for *piece-wise* polynomial bounds,
- zero-order types: unnanotated lists, sized integers and beyond matrices $\mathrm{L}_n(\mathrm{L}^{\exists\, i\ .Q(n,i)}_{p(n,i)}(-))$,
- algebraic data structures,
- the infinite cylinders issue.

Summary

- a polynomial-size-annotated type system is designed,
- checking is decidable in reals and is, basically, adjusted for integers,
- test-based inference of polynomial lower and upper bounds is possible.

Future work:

- test-based inference for *piece-wise* polynomial bounds,
- zero-order types: unnanotated lists, sized integers and beyond matrices $L_n(L_{p(n,i)}^{\exists\, i\ .Q(n,i)}(-))$,
- algebraic data structures,
- the infinite cylinders issue.

Summary

- a polynomial-size-annotated type system is designed,
- checking is decidable in reals and is, basically, adjusted for integers,
- test-based inference of polynomial lower and upper bounds is possible.

Future work:

- test-based inference for *piece-wise* polynomial bounds,
- zero-order types: unnanotated lists, sized integers and beyond matrices $\mathrm{L}_n(\mathrm{L}^{\exists\, i\ .Q(n,i)}_{p(n,i)}(-))$,
- algebraic data structures,
- the infinite cylinders issue.

Summary

- a polynomial-size-annotated type system is designed,
- checking is decidable in reals and is, basically, adjusted for integers,
- test-based inference of polynomial lower and upper bounds is possible.

Future work:

- test-based inference for *piece-wise* polynomial bounds,
- zero-order types: unnanotated lists, sized integers and beyond matrices $\mathrm{L}_n(\mathrm{L}^{\exists\, i\ .Q(n,i)}_{p(n,i)}(-))$,
- algebraic data structures,
- the infinite cylinders issue.