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.

・ 同 ト ・ ヨ ト ・ ヨ ト ・

Outline



- Size information for memory/time management
- Our previous work: strict polynomial size dependencies
- Our results: lower and upper bounds for non-monotone dependencies
 - A language and its type system
 - Type-checking decidable in reals
 - Test-based inference

・ 同 ト ・ ヨ ト ・ ヨ ト ・

Motivation

Our results: lower and upper bounds for non-monotone dependence Summary Size information for memory/time management Our previous work: strict polynomial size dependencies

イロト イポト イヨト イヨト

Outline



- Size information for memory/time management
- Our previous work: strict polynomial size dependencies
- 2 Our results: lower and upper bounds for non-monotone dependencies
 - A language and its type system
 - Type-checking decidable in reals
 - Test-based inference

Size information for memory/time management Our previous work: strict polynomial size dependencies

ヘロト ヘ回ト ヘヨト ヘヨト

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.

Motivation

Our results: lower and upper bounds for non-monotone dependence Summary Size information for memory/time management Our previous work: strict polynomial size dependencies

イロト イポト イヨト イヨト

Outline



- Size information for memory/time management
- Our previous work: strict polynomial size dependencies
- 2 Our results: lower and upper bounds for non-monotone dependencies
 - A language and its type system
 - Type-checking decidable in reals
 - Test-based inference

ヘロト 人間 とくほとくほとう

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 : L_n(α) × L_m(α) → L_{n+m}(α) sqdiff : L_n(α) × L_m(α) → L_{(n-m)²}(α)
- 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:

 $\begin{array}{l} \text{insert'}: \ \text{Int} \times L_n(\text{Int}) \rightarrow L_{n,\,n+1}(\text{Int}) \\ \text{delete'}: \ \text{Int} \times L_n(\text{Int}) \rightarrow L_{n,\,n+1}(\text{Int}) \end{array}$

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:

 $\begin{array}{l} \text{insert': } \text{Int} \times L_n(\text{Int}) \rightarrow L_{n,\,n+1}(\text{Int}) \\ \text{delete': } \text{Int} \times L_n(\text{Int}) \rightarrow L_{n,\,n+1}(\text{Int}) \end{array}$

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)





- Size information for memory/time management
- Our previous work: strict polynomial size dependencies
- Our results: lower and upper bounds for non-monotone dependencies
 - A language and its type system
 - Type-checking decidable in reals

イロト イポト イヨト イヨト

Motivation	A language and its type system
Our results: lower and upper bounds for non-monotone dependence	Type-checking decidable in reals
Summary	Test-based inference

The language

$$\begin{array}{rcl} \textit{Basic} & b & ::= & c \mid \mathsf{Nil} \mid \mathsf{Cons}(x,\,y) \mid f(x_1,\ldots,x_n) \\ \textit{Expr} & e & ::= & \mathsf{letfun} \ f(x_1,\ldots,x_n) = e_1 \ \mathsf{in} \ e_2 \\ & \mid b \\ & \mid \mathsf{let} \ x = b \ \mathsf{in} \ e \\ & \mid \mathsf{if} \ x \ \mathsf{then} \ e_1 \ \mathsf{else} \ e_2 \\ & \mid \mathsf{match} \ x \ \mathsf{with} \mid \mathsf{Nil} \Rightarrow e_1 \\ & \mid \mathsf{Cons}(\mathsf{hd},\,\mathsf{tl}) \Rightarrow e_2 \end{array}$$

・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・

 Motivation
 A language and its type system

 Our results: lower and upper bounds for non-monotone dependen
 Type-checking decidable in reals

 Summary
 Test-based inference

Zero-order types

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



イロン 不得 とくほ とくほう 一座

 Motivation
 A language and its type system

 Our results: lower and upper bounds for non-monotone dependen
 Type-checking decidable in reals

 Summary
 Test-based inference

Zero-order types

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



ヘロア ヘビア ヘビア・

Motivation	A language and its type system
Our results: lower and upper bounds for non-monotone dependenc	Type-checking decidable in reals
Summary	Test-based inference

Zero-order types

An example of types with size/type variables:

 $L_n(\alpha) \qquad (formal-parameter types for functions) \\ L_{\alpha^2+i}^{0 \le i \le n}(\alpha) \qquad (output types for functions)$

Types
$$\tau$$
 ::= Int |Bool | α | $L^{P(\overline{n},\overline{i})}_{\rho(\overline{n},\overline{i})}(\tau)$,

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 -)

Semantics in an example: $x : L_{m \to j}^{0 \le j \le n} (L_{mn+i}^{0 \le i \le n^2}(\alpha)) \mid \exists \ 0 \le j \le n, \ 0 \le i \le n^2$ $x : L_{m \to j} (L_{mn+i}(\alpha))$

イロン 不良 とくほう 不良 とうほ

Motivation	A language and its type system
Our results: lower and upper bounds for non-monotone dependenc	Type-checking decidable in reals
Summary	Test-based inference

Zero-order types

An example of types with size/type variables:

 $L_n(\alpha) \qquad (formal-parameter types for functions) \\ L_{n^2+i}^{0 \le i \le n}(\alpha) \qquad (output types for functions)$

Types
$$\tau$$
 ::= Int |Bool | α | $L_{\rho(\overline{n},\overline{i})}^{P(\overline{n},i)}(\tau)$,

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 -)

Semantics in an example:

$$x : L_{m-j}^{0 \le j \le n} (L_{mn+i}^{0 \le i \le n^2}(\alpha)) \mid \exists \ 0 \le j \le n, \ 0 \le i \le n^2 \\ x : L_{m-j} (L_{mn+i}(\alpha))$$

イロト イポト イヨト イヨト 一座

Motivation	A language and its type system
Our results: lower and upper bounds for non-monotone dependence	Type-checking decidable in reals
Summary	lest-based inference

insert	: $(\alpha \times \alpha \to Bool) \times \alpha \times L_n(\alpha)$	\rightarrow	$L_{n+i}^{0\leq i\leq 1}(\alpha)$
rinsert	: $(\alpha \times \alpha \to Bool) \times L_{\mathbf{n}}(\alpha) \times L_{\mathbf{m}}(\alpha)$	\rightarrow	$L_{m+i}^{0 \leq i \leq n}(\alpha)$
filter	: $(\alpha \times \alpha \rightarrow Bool) \times L_{n}(\alpha)$	\longrightarrow	$L^{0 \leq i \leq n}_i(\alpha)$
delete	: $(\alpha \times \alpha \to Bool) \times \alpha \times L_{n}(\alpha)$	\longrightarrow	$L_{n \neq i}^{0 \leq i \leq 1}(\alpha)$
rdelete	: $(\alpha \times \alpha \to Bool) \times L_{\mathbf{n}}(\alpha) \times L_{\mathbf{m}}(\alpha)$	\longrightarrow	$L_{m \neq i}^{0 \leq i \leq n}(\alpha)$
divtwo	: L _n (α)	\rightarrow	$L^{\underline{0\leq i\leq 1}}_{\frac{\underline{n+i}}{2}}(\alpha)$

- 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.

・ 同 ト ・ ヨ ト ・ ヨ ト

Motivation	A language and its type syst
Our results: lower and upper bounds for non-monotone dependence	Type-checking decidable in r
Summary	Test-based inference

insert	: $(\alpha \times \alpha \to Bool) \times \alpha \times L_{n}(\alpha)$	\rightarrow	$L_{n+i}^{0\leq i\leq 1}(\alpha)$
rinsert	: $(\alpha \times \alpha \to Bool) \times L_{n}(\alpha) \times L_{m}(\alpha)$	\rightarrow	$L_{m+i}^{0 \leq i \leq n}(\alpha)$
filter	: $(\alpha \times \alpha \rightarrow Bool) \times L_{n}(\alpha)$	\rightarrow	$L_{i}^{0 \leq i \leq n}(\alpha)$
delete	: $(\alpha \times \alpha \to Bool) \times \alpha \times L_{n}(\alpha)$	\longrightarrow	$L_{n \doteq i}^{0 \leq i \leq 1}(\alpha)$
rdelete	: $(\alpha \times \alpha \to Bool) \times L_{n}(\alpha) \times L_{m}(\alpha)$	\longrightarrow	$L_{m \doteq i}^{0 \le i \le n}(\alpha)$
divtwo	: L _n (α)	\longrightarrow	$L^{\underline{0}\leq i\leq 1}_{\frac{\underline{n}\neq i}{2}}(\alpha)$

em

・ 同 ト ・ 三 ト ・

.≣⇒

- 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.

Motivation	A language and its t
Our results: lower and upper bounds for non-monotone dependenc	Type-checking decid
Summary	Test-based inference

insert	: $(\alpha \times \alpha \to Bool) \times \alpha \times L_{n}(\alpha)$	\rightarrow	$L_{n+i}^{0\leq i\leq 1}(\alpha)$
rinsert	: $(\alpha \times \alpha \to Bool) \times L_{n}(\alpha) \times L_{m}(\alpha)$	\rightarrow	$L_{m+i}^{0 \leq i \leq n}(\alpha)$
filter	: $(\alpha \times \alpha \rightarrow Bool) \times L_{n}(\alpha)$	\rightarrow	$L_i^{0 \le i \le n}(\alpha)$
delete	: $(\alpha \times \alpha \to Bool) \times \alpha \times L_{n}(\alpha)$	\rightarrow	$L_{n \doteq i}^{0 \leq i \leq 1}(\alpha)$
rdelete	: $(\alpha \times \alpha \to Bool) \times L_{n}(\alpha) \times L_{m}(\alpha)$	\rightarrow	$L_{m \doteq i}^{0 \leq i \leq n}(\alpha)$
divtwo	: L _n (α)	\longrightarrow	$L^{0\leq i\leq 1}_{\frac{n+i}{2}}(\alpha)$

pe system

ヘロト ヘ戸ト ヘヨト ヘヨト

- 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.

Motivation	A language and its typ
Our results: lower and upper bounds for non-monotone dependence	Type-checking decidal
Summary	Test-based inference

insert	: $(\alpha \times \alpha \to Bool) \times \alpha \times L_{n}(\alpha)$	\rightarrow	$L_{n+i}^{0\leq i\leq 1}(\alpha)$
rinsert	: $(\alpha \times \alpha \to Bool) \times L_{\mathbf{n}}(\alpha) \times L_{\mathbf{m}}(\alpha)$	\rightarrow	$L_{m+i}^{0 \leq i \leq n}(\alpha)$
filter	: $(\alpha \times \alpha \rightarrow Bool) \times L_{n}(\alpha)$	\rightarrow	$L_i^{0 \le i \le n}(\alpha)$
delete	: $(\alpha \times \alpha \to Bool) \times \alpha \times L_{n}(\alpha)$	\rightarrow	$L_{n \doteq i}^{0 \leq i \leq 1}(\alpha)$
rdelete	: $(\alpha \times \alpha \to Bool) \times L_{\mathbf{n}}(\alpha) \times L_{\mathbf{m}}(\alpha)$	\rightarrow	$L_{m \doteq i}^{0 \leq i \leq n}(\alpha)$
divtwo	: L <mark>n</mark> (α)	\rightarrow	$L_{n-i}^{0\leq i\leq 1}(\alpha)$
			2

e system

(本間) (本語) (本語)

- 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.

Motivation	A language and its type
Our results: lower and upper bounds for non-monotone dependence	Type-checking decidab
Summary	Test-based inference

insert	: $(\alpha \times \alpha \to Bool) \times \alpha \times L_n(\alpha)$	\rightarrow	$L_{n+i}^{0\leq i\leq 1}(\alpha)$
rinsert	: $(\alpha \times \alpha \to Bool) \times L_{\mathbf{n}}(\alpha) \times L_{\mathbf{m}}(\alpha)$	\rightarrow	$L_{m+i}^{0 \leq i \leq n}(\alpha)$
filter	: $(\alpha \times \alpha \rightarrow Bool) \times L_{n}(\alpha)$	\rightarrow	$L_i^{0 \le i \le n}(\alpha)$
delete	: $(\alpha \times \alpha \to Bool) \times \alpha \times L_{n}(\alpha)$	\rightarrow	$L_{n \doteq i}^{0 \leq i \leq 1}(\alpha)$
rdelete	: $(\alpha \times \alpha \to Bool) \times L_{\mathbf{n}}(\alpha) \times L_{\mathbf{m}}(\alpha)$	\rightarrow	$L_{m \doteq i}^{0 \leq i \leq n}(\alpha)$
divtwo	: L <mark>n</mark> (α)	\rightarrow	$L_{n-i}^{0\leq i\leq 1}(\alpha)$
			2

system

・ 同 ト ・ ヨ ト ・ ヨ ト

- 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.

Motivation	A language and its type system
Our results: lower and upper bounds for non-monotone dependenc	Type-checking decidable in reals
Summary	Test-based inference

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

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

◆□ ▶ ◆□ ▶ ◆ □ ▶ ◆ □ ▶ ● □ ● ● ● ●

 $\begin{array}{c|c} D(\overline{n},\overline{j}) \vdash & \\ \mathsf{L}^{Q(\overline{n},\overline{j})}_{p(\overline{n},\overline{l})}(\tau) \triangleleft \mathsf{L}^{Q'(\overline{n},\overline{j}')}_{p'(\overline{n},\overline{j}')+1}(\tau') & \forall \ \overline{n}\overline{j}\overline{j}' \ \exists \overline{l}. \ D(\overline{n},\overline{j}) \land Q'(\overline{n},\overline{j}') \Rightarrow \\ Q(\overline{n},\overline{l}) \land p(\overline{n},\overline{l}) = p'(\overline{n},\overline{j}') + 1 \end{array}$

Motivation	A language and its type system
Dur results: lower and upper bounds for non-monotone dependenc	Type-checking decidable in reals
Summary	Test-based inference

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

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

◆□ ▶ ◆□ ▶ ◆ □ ▶ ◆ □ ▶ ● □ ● ● ● ●

 $\begin{array}{c|c} D(\overline{n},\overline{j}) \vdash & \\ \mathsf{L}^{Q(\overline{n},\overline{j})}_{p(\overline{n},\overline{i})}(\tau) \triangleleft \mathsf{L}^{Q'(\overline{n},\overline{j}')}_{p'(\overline{n},\overline{j}')+1}(\tau') & \forall \ \overline{n}\overline{j}\overline{j}' \ \exists \overline{i}. \ D(\overline{n},\overline{j}) \land Q'(\overline{n},\overline{j}') \Rightarrow \\ Q(\overline{n},\overline{i}) \land p(\overline{n},\overline{i}) = p'(\overline{n},\overline{j}') + 1 \end{array}$

Motivation	A language and its type system
Dur results: lower and upper bounds for non-monotone dependenc	Type-checking decidable in reals
Summary	Test-based inference

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

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

◆□ > ◆□ > ◆目 > ◆目 > ● ● ● ●

$$\begin{array}{c|c} D(\overline{n},\overline{j}) \vdash & \\ \mathsf{L}^{Q(\overline{n},\overline{i})}_{p(\overline{n},\overline{i})}(\tau) \triangleleft \mathsf{L}^{Q'(\overline{n},\overline{j}')}_{p'(\overline{n},\overline{j}')+1}(\tau') & \forall \ \overline{n}\overline{j}\overline{j}' \ \exists \overline{i}. \ D(\overline{n},\overline{j}) \land Q'(\overline{n},\overline{j}') \Rightarrow \\ Q(\overline{n},\overline{i}) \land p(\overline{n},\overline{i}) = p'(\overline{n},\overline{j}') + 1 \end{array}$$

Motivation	A language and its type system
Our results: lower and upper bounds for non-monotone dependence	Type-checking decidable in reals
Summary	Test-based inference

$$\begin{split} \Sigma(f) &= \tau_1^f \times \ldots \times \tau_{k'}^f \times \tau_1^\circ \times \ldots \times \tau_k^\circ \to \tau_0 \\ \Sigma(g_1) &= \tau_1^f, \ldots, \Sigma(g_{k'}) = \tau_{k'}^f \\ \frac{D \vdash \tau_0' \triangleleft \ast(\tau_0)}{D; \ \Gamma, x_1 : \tau_1', \ldots, x_1 : \tau_k' \vdash_{\Sigma} f(g_1, \ldots, g_{k'}, \ x_1, \ldots, x_k) : \tau_0'} \ \mathsf{FAPP} \end{split}$$

$$\begin{array}{rcl} & \text{Substitution }* \text{ on free size parameters} \\ \text{input} & \mathsf{L}_{m}(-) & \mapsto & \mathsf{L}_{\rho(\overline{n},\,\overline{i})}^{P(\overline{n},\,\overline{i})}(-) \\ \text{output} & \mathsf{L}_{q(...,\,m,\,...,\,\overline{j})}^{Q(...,\,m,\,...,\,\overline{j})}() \mapsto & \mathsf{L}_{q(...,\,p(\overline{n},\,\overline{i}),...,\,\overline{j})}^{P(\overline{n},\,\overline{i}), Q(...,\,p(\overline{n},\,\overline{i}),...,\,\overline{j})}(-) \end{array}$$

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

ヘロン 人間 とくほど 人ほど 一座

Motivation	A language and its type system
Our results: lower and upper bounds for non-monotone dependence	Type-checking decidable in reals
Summary	Test-based inference

$$\begin{split} \Sigma(f) &= \tau_1^f \times \ldots \times \tau_{k'}^f \times \tau_1^\circ \times \ldots \times \tau_k^\circ \to \tau_0 \\ \Sigma(g_1) &= \tau_1^f, \ldots, \Sigma(g_{k'}) = \tau_{k'}^f \\ \hline D \vdash \tau_0' \triangleleft *(\tau_0) \\ \hline D; \ \Gamma, x_1 \colon \tau_1', \ldots, x_1 \colon \tau_k' \vdash_{\Sigma} f(g_1, \ldots, g_{k'}, x_1, \ldots, x_k) \colon \tau_0' \end{split} \mathsf{FAPP} \end{split}$$

Substitution * on free size parameters

input
$$L_m(-) \mapsto L_{p(\overline{n},\overline{i})}^{P(n,i)}(-)$$

output $L_{q(\dots,m,\dots,\overline{j})}^{Q(\dots,m,\dots,\overline{j})}(-) \mapsto L_{q(\dots,p(\overline{n},\overline{i}),\dots,\overline{j})}^{P(\overline{n},\overline{i})}(-)$

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

・ロト ・ 同ト ・ ヨト ・ ヨト - 三日



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

$$\begin{aligned} \Gamma(\mathbf{x}) &= \mathsf{Bool} \\ D; \ \Gamma \vdash_{\Sigma} \boldsymbol{e}_{t} : \tau \\ D; \ \Gamma \vdash_{\Sigma} \boldsymbol{e}_{f} : \tau \\ \hline D; \ \Gamma \vdash_{\Sigma} if x \text{ then } \boldsymbol{e}_{t} \text{ else } \boldsymbol{e}_{f} : \tau \end{aligned}$$
 IF

くロト (過) (目) (日)

2







- Size information for memory/time management
- Our previous work: strict polynomial size dependencies

Our results: lower and upper bounds for non-monotone dependencies

- A language and its type system
- Type-checking decidable in reals
- Test-based inference

イロト イポト イヨト イヨト

A language and its type system Type-checking decidable in reals Test-based inference

1

æ

イロト 不得 とくほと くほとう

Type checking: example

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

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

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

A language and its type system Type-checking decidable in reals Test-based inference

◆□ > ◆□ > ◆臣 > ◆臣 > ─臣 ─のへで

Type checking: example

$$n = 0$$
 \vdash $n + ?i = 0 \land 0 \leq ?i \leq 1$

$$n > 0$$
 \vdash $n + ?i = n \land 0 \leq ?i \leq 1$

$$n > 0; \ 0 \le j \le 1 \quad \vdash \quad n + ?i = (n - 1) + j + 1 \land 0 \le ?i \le 1$$

Solution:

$$\begin{array}{ccc} & \vdash & ?i := 0 \land 0 \le ?i \le 1 \\ n > 0 & \vdash & ?i := n - n = 0 \land 0 \le ?i \le 1 \\ n > 0; \ 0 \le j \le 1 & \vdash & ?i := j \land 0 \le ?i \le 1 \end{array}$$

A language and its type system Type-checking decidable in reals Test-based inference

ヘロト 人間ト 人団ト 人団ト

Output size annotations

$$\begin{array}{rll} \text{rinsert} & : (\alpha \times \alpha \to \mathsf{Bool}) \times \mathsf{L}_{\mathbf{n}}(\alpha) \times \mathsf{L}_{\mathbf{m}}(\alpha) & \to & \mathsf{L}_{m+i}^{0 \leq i \leq n}(\alpha) \\ \text{rdelete} & : (\alpha \times \alpha \to \mathsf{Bool}) \times \mathsf{L}_{\mathbf{n}}(\alpha) \times \mathsf{L}_{\mathbf{m}}(\alpha) & \to & \mathsf{L}_{m-i}^{0 \leq i \leq n}(\alpha) \end{array}$$



This

 makes type checking easier, since end entailments are of the form

- $D(\overline{n},\overline{j}) \vdash p(\overline{n}) + i = q(\overline{n},\overline{j}) \land Q(\overline{n},i)$ i.e. to check $D(\overline{n},\overline{j}) \vdash Q(\overline{n},q(\overline{n},\overline{j})-p(\overline{n}))$
- or D(n,j) ⊢ p(n) − i = q(n,j) ∧ Q(n,i)
 i.e. to check ... see the next slide ...

comes from one natural observation (see the next slides).

A language and its type system Type-checking decidable in reals Test-based inference

・ 同 ト ・ ヨ ト ・ ヨ ト

Output size annotations

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

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

This

- makes type checking easier, since end entailments are of the form
 - $D(\overline{n},\overline{j}) \vdash p(\overline{n}) + i = q(\overline{n},\overline{j}) \land Q(\overline{n},i)$ i.e. to check $D(\overline{n},\overline{j}) \vdash Q(\overline{n},q(\overline{n},\overline{j}) - p(\overline{n}))$,
 - or D(n,j) ⊢ p(n) − i = q(n,j) ∧ Q(n,i)
 i.e. to check ... see the next slide ...

2 comes from one natural observation (see the next slides).

A language and its type system **Type-checking decidable in reals** Test-based inference

Output size annotations

Size annot. of the form $Q(\overline{n}, i)$ of the form

 $p(\overline{n}) + i \text{ or } p(\overline{n}) - i$ $0 \le i \le \delta(\overline{n})$

This

- makes type checking easier, since end entailments are of the form
 - (see above),
 - or $D(\overline{n},\overline{j}) \vdash p(\overline{n}) i = q(\overline{n},\overline{j}) \land Q(\overline{n},i)$
 - e. to check one of the $D(\overline{n},\overline{j}), p(\overline{n}) q(\overline{n},\overline{j}) \le p(\overline{n}) \vdash Q(\overline{n},p(\overline{n}) q(\overline{n},\overline{j}))$ $D(\overline{n},\overline{j}), i > p(\overline{n}) \vdash q(\overline{n},\overline{j}) = 0 \land Q(\overline{n},i)$

2 comes from one natural observation (see the next slide)

A language and its type system **Type-checking decidable in reals** Test-based inference

Output size annotations

Size annot. of the form $Q(\overline{n}, i)$ of the form

 $p(\overline{n}) + i \text{ or } p(\overline{n}) - i$ $0 \le i \le \delta(\overline{n})$

イロト 不得 トイヨト イヨト

This

- makes type checking easier, since end entailments are of the form
 - (see above),
 - or $D(\overline{n},\overline{j}) \vdash p(\overline{n}) i = q(\overline{n},\overline{j}) \land Q(\overline{n},i)$ i.e. to check one of the $D(\overline{n},\overline{j}), p(\overline{n}) - q(\overline{n},\overline{j}) \le p(\overline{n}) \vdash Q(\overline{n}, p(\overline{n}) - q(\overline{n},\overline{j}))$, $D(\overline{n},\overline{j}), i > p(\overline{n}) \mapsto q(\overline{n},\overline{j}) = 0 \land Q(\overline{n},i)$

2 comes from one natural observation (see the next slide)

A language and its type system Type-checking decidable in reals Test-based inference

Output size annotations

Size annot. of the form
$Q(\overline{n}, i)$ of the form

 $p(\overline{n}) + i \text{ or } p(\overline{n}) - i$ $0 \le i \le \delta(\overline{n})$

<ロ> (四) (四) (三) (三) (三)

This

- makes type checking easier,
- 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 $0 \le i \le p_{max}(\overline{n}) - p_{min}(\overline{n})$ or $p_{max}(\overline{n}) - i$ for some $0 \le i \le p_{max}(\overline{n}) - p_{min}(\overline{n})$

A language and its type system Type-checking decidable in reals Test-based inference

ヘロン 人間 とくほ とくほ とう

Checking using reals (CAD)

Real arithmetic is inevitable.

Just embedding integers into reals is not enough! E.g. $x^2 \le x^3$ 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 $(\overline{n}, \overline{j})$ $D(\overline{n}, \overline{j}) \land \neg Q(\overline{n}, \overline{j})$

A language and its type system Type-checking decidable in reals Test-based inference

・ロン ・ 一 と ・ 日 と ・ 日 と

Checking using reals (CAD)

Real arithmetic is inevitable.

Just embedding integers into reals is not enough! E.g. $x^2 \le x^3$ 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 $(\overline{n}, \overline{j})$ $D(\overline{n}, \overline{j}) \land \neg Q(\overline{n}, \overline{j})$

A language and its type system Type-checking decidable in reals Test-based inference

・ロト ・ 同ト ・ ヨト ・ ヨト - 三日

Checking using reals (CAD)

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

 $g_{11} \leq x_1 \leq g_{12} \ g_{21}(x_1) \leq x_2 \leq g_{22}(x_1)$

where g_{ij} contains +, -, * and radicals.

The question: are there integer numbers in the CAD for $D(\overline{n}, \overline{j}) \wedge \neg Q(\overline{n}, \overline{j})$? 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: $g_{12} = \infty$.

A language and its type system Type-checking decidable in reals Test-based inference

<ロ> (四) (四) (三) (三) (三)

Checking using reals (CAD)

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

 $g_{11} \leq x_1 \leq g_{12}$ $g_{21}(x_1) \leq x_2 \leq g_{22}(x_1)$

where g_{ij} contains +, -, * and radicals.

The question: are there integer numbers in the CAD for $D(\overline{n}, \overline{j}) \wedge \neg Q(\overline{n}, \overline{j})$? 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: $g_{12} = \infty$.

A language and its type system Type-checking decidable in reals Test-based inference

イロン 不良 とくほう 不良 とうほ

Checking using reals (CAD)

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

 $g_{11} \leq x_1 \leq g_{12}$ $g_{21}(x_1) \leq x_2 \leq g_{22}(x_1)$...

where g_{ij} contains +, -, * and radicals.

The question: are there integer numbers in the CAD for $D(\overline{n}, \overline{j}) \wedge \neg Q(\overline{n}, \overline{j})$? 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: $g_{12} = \infty$.

Mic	otivation	A language and its type system
Our results: lower and upper bounds for non-monotone	dependenc	Type-checking decidable in reals
Si	ummary	Test-based inference



Outline

- Motivation
- Size information for memory/time management
- Our previous work: strict polynomial size dependencies

Our results: lower and upper bounds for non-monotone dependencies

- A language and its type system
- Type-checking decidable in reals
- Test-based inference

・ 同 ト ・ ヨ ト ・ ヨ ト

A language and its type system Type-checking decidable in reals Test-based inference

イロト イポト イヨト イヨト 三日

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: $p(0) \rightarrow 1$ $p(n) \rightarrow n \mid 1 + p(n-1)$

A language and its type system Type-checking decidable in reals Test-based inference

イロト イポト イヨト イヨト 三日

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: $p(0) \rightarrow 1$ $p(n) \rightarrow n \mid 1 + p(n-1)$

A language and its type system Type-checking decidable in reals Test-based inference

イロン 不良 とくほう 不良 とうほ

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: $p(0) \rightarrow 1$ $p(n) \rightarrow n \mid 1 + p(n-1)$ Motivation A language and its type system Our results: lower and upper bounds for non-monotone dependenc Summary Test-based inference

Inference via abstract "testing"

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

$$\begin{array}{c} p(1) = \{1, 2\} \\ p(2) = \{2, 3\} \end{array} \} \Rightarrow \begin{cases} p_{min}(1) = 1, \ p_{min}(2) = 2 \\ p_{max}(1) = 2, \ p_{max}(2) = 3 \end{cases}$$

Solving the system of linear equation gives

$$p_{min}(n) = n$$

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

<ロ> (四) (四) (三) (三) (三)

Motivation A language and its type syster Our results: lower and upper bounds for non-monotone dependenc Summary Test-based inference

Inference via abstract "testing"

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

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

Solving the system of linear equation gives

$$p_{min}(n) = n$$

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

<ロ> (四) (四) (三) (三) (三)

- 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^{∃ i .Q(n, i)}_{p(n, i)}(−)),
- algebraic data structures,
- the infinite cylinders issue.

イロト イポト イヨト イヨト 三油

- 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^{∃ i}, Q(n, i)</sup>_{p(n, i)}(−)),
- algebraic data structures,
- the infinite cylinders issue.

イロト イポト イヨト イヨト 三油

- 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^{∃ i.Q(n, i)}_{p(n, i)}(−)),
- algebraic data structures,
- the infinite cylinders issue.

イロン 不良 とくほう 不良 とうほ

- 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^{∃ i.Q(n, i)}_{p(n, i)}(−)),
- algebraic data structures,
- the infinite cylinders issue.

ヘロン 人間 とくほ とくほ とう