Termination Checking = Type Checking

Structural Recursion beyond Subterm Ordering

Andreas Abel

International Summer School Marktoberdorf
August 3, 2001

Slide 1

1. Introduction
2. Informal Description
3. Properties
4. Applications and Further Work

Work supported by: Office for Technology in Education (Carnegie Mellon, Pittsburgh), PhD Programme Logic in Computer Science (GKLI, Munich)

Slide 2

Termination Checking

- Special case: Structural recursion (examples follow!)
- Applications:
  - Functional and logic programming
  - Theorem proving: proof as recursive function (LEGO, Coq, Twelf, . . .)
- Methods:
  - Subterm ordering
  - Types (*New!*)

Example: Binary Logarithm

Slide 3

$$
\begin{aligned}
&\mathsf{half} : \mathsf{nat} \to \mathsf{nat}\\
&\mathsf{half}\ 0 = 0\\
&\mathsf{half}\ (\mathsf{s}\,0) = (\mathsf{s}\,0)\\
&\mathsf{half}\ (\mathsf{s}\,(\mathsf{s}\,(x))) = (\mathsf{s}\,(\mathsf{half}\,x))\\
&\mathsf{log} : \mathsf{nat} \to \mathsf{nat}\\
&\mathsf{log}\ 0 = 0\\
&\mathsf{log}\ (\mathsf{s}\,0) = 0\\
&\mathsf{log}\ (\mathsf{s}\,(\mathsf{s}\,(x))) = (\mathsf{s}\,(\mathsf{log}\,(\mathsf{s}\,(\mathsf{half}\,x))))
\end{aligned}
$$

Central Idea I: Approximation Types

Slide 4

- Recursion over *inductive datatypes*: nat, list, binary trees, first-order terms, ...
- Inhabitants can be viewed as trees.
- *Approximation* $Y \leq$ list: lists upto a fixed but unknown maximal length n.
- *Decoration* Y^i: lists upto length $n + i$.
- Notation: $Y^+ = Y^1$, $Y^- = Y^{-1}$, ...
- *Subtyping*: $Y^i \leq Y^j \leq$ list for $i \leq j$.
- Function
  $$g(x^{Y^+}) = \ldots g(M) \ldots$$
  terminates if all arguments M of recursive calls are of type Y.

Slide 5

Central Idea II: Size-Analysis

- *Bounded quantification*:

$$\mathsf{inssort} : \forall Y \leq \mathsf{list}\,.\, Y \to \mathsf{list}$$

- Function defined for all approximations.
- Use of type variables gives simple *size-analysis for free!*

$$\mathsf{inssort} : \forall Y \leq \mathsf{list}\,.\, Y \to Y$$

- Meaning: Size of output of the same maximal length than input.

Slide 6

Example: Binary Logarithm

$$\begin{aligned}
&\mathsf{half} : \mathsf{nat} \to \mathsf{nat}\\
&\mathsf{half}\ 0 = 0\\
&\mathsf{half}\ (\mathsf{s}\,0) = (\mathsf{s}\,0)\\
&\mathsf{half}\ (\mathsf{s}\,(\mathsf{s}\,(x\ \ )\ )\ \ ) = (\mathsf{s}\,(\mathsf{half}\ x)\ \ )
\end{aligned}$$

$$\begin{aligned}
&\mathsf{log} : \mathsf{nat} \to \mathsf{nat}\\
&\mathsf{log}\ 0 = 0\\
&\mathsf{log}\ (\mathsf{s}\,0) = 0\\
&\mathsf{log}\ (\mathsf{s}\,(\mathsf{s}\,(x\ \ )\ )\ \ ) = (\mathsf{s}\,(\mathsf{log}\,(\mathsf{s}\,(\mathsf{half}\ x)\ \ )\ ))
\end{aligned}$$

Example: Binary Logarithm

Slide 7

$$
\begin{aligned}
&\mathsf{half} : \forall Y \leq \mathsf{nat}.\ Y \to Y \\
&\mathsf{half}\ 0^{Y^+} &&= 0^{Y^+} \\
&\mathsf{half}\ (\mathsf{s}\,0)^{Y^+} &&= (\mathsf{s}\,0)^{Y^+} \\
&\mathsf{half}\ (\mathsf{s}\,(\mathsf{s}\,(x^{Y^-})^{Y})^{Y^+}) &&= (\mathsf{s}\,(\mathsf{half}\,x)^{Y^-})^{Y \leq Y^+} \\
&\mathsf{log} : \forall Y \leq \mathsf{nat}.\ Y \to Y \\
&\mathsf{log}\ 0^{Y^+} &&= 0^{Y^+} \\
&\mathsf{log}\ (\mathsf{s}\,0)^{Y^+} &&= 0^{Y \leq Y^+} \\
&\mathsf{log}\ (\mathsf{s}\,(\mathsf{s}\,(x^{Y^-})^{Y})^{Y^+}) &&= (\mathsf{s}\,(\mathsf{log}\,(\mathsf{s}\,(\mathsf{half}\,x)^{Y^-})^{Y}))^{Y^+}
\end{aligned}
$$

Example: Insertion Sort

Slide 8

$$
\begin{aligned}
&\mathsf{insert} : \mathsf{int} \to \forall Y \leq \mathsf{int\ list}.\, Y \to Y^+ \\
&\mathsf{insert}\ a\ []^{Y^+} &&= [a]^{Y^{++}} \\
&\mathsf{insert}\ a\ (x :: xs^{Y})^{Y^+} &&= \mathsf{if}\ a > x \\
& &&\quad \mathsf{then}\ (x :: (\mathsf{insert}\ a\ xs)^{Y^+})^{Y^{++}} \\
& &&\quad \mathsf{else}\ (a :: x :: xs^{Y})^{Y^{++}} \\
&\mathsf{inssort} : \forall Y \leq \mathsf{int\ list}.\, Y \to Y \\
&\mathsf{inssort}\ ([])^{Y^+} &&= ([])^{Y^+} \\
&\mathsf{inssort}\ ([x])^{Y^+} &&= ([x])^{Y^+} \\
&\mathsf{inssort}\ (x :: xs^{Y})^{Y^+} &&= (\mathsf{insert}\ x\ (\mathsf{inssort}\ xs)^{Y})^{Y^+}
\end{aligned}
$$

System and Properties

Slide 9

- All inductive datatypes are *constructed* from primitive type constructors.
- Type system includes a lean subtyping calculus for approximations.
- Type and termination checking and size analysis in *one pass*.
- Properties:
  1. Type preservation *(done)*
  2. Weak normalization *(easy, if you know how to)*
  3. *Strong normalization* *(work in progress)*

Applications

Slide 10

- Functional programming: mix with partial functions.
- Theorem proving [Gimenéz, 1998] .
- Partial evaluation.
- Rich type systems (e.g. resource bound certification [Crary/Weirich, 2000]) .

Further Work

Slide 11

- Polymorphism, constructors (F^ω).
- More orders: lexicographic etc.
- Mutual recursion [Abel/Altenkirch, 2001] , [Lee/Jones/Ben-Amram, 2001].
- Coinductive types [Gimenéz, 1998].

References

Slide 12

[1] Andreas Abel and Thorsten Altenkirch. A predicative analysis of structural recursion. To appear in the Journal of Functional Programming, June 2001.

[2] Karl Crary and Stephanie Weirich. Resource bound certification. In *Proceedings of the 27th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 184–198, Boston, Massachusetts, USA, January 2000.

[3] Eduardo Gimenéz. Structural recursive definitions in type theory. In *Automata, Languages and Programming, 25th International Colloquium, ICALP'98, Aalborg, Denmark, July 13-17, 1998, Proceedings*, volume 1443 of *LNCS*, pages 397–408. Springer, 1998.

[4] Chin Soon Lee, Neil D. Jones, and Amir M. Ben-Amram. The size-change principle for program termination. In *ACM Symposium on Principles of Programming Languages (POPL'01)*, London, UK, January 2001. ACM Press.