# Termination of Functions that Are Passed to Their Arguments

Andreas Abel
Dept. of Comp. Sci., Chalmers

**Slide 1**

September 13, 2005
APPSEM II Workshop
Frauenchiemsee, Bavaria

Work supported by: GKLI (DFG), TYPES, APPSEM-II and CoVer (SSF)

## Quiz: Is `eqList` terminating on all total inputs?

**Slide 2**

```
data  MList m a  where
    Nil  :: MList m a
    Cons :: a -> m (MList m a) -> MList m a

eqList eqM eq Nil Nil = True

eqList eqM eq (Cons a mas) (Cons b mbs)
                      = eq a b
                        && eqM (eqList eqM eq) mas mbs

eqList eqM eq _ _     = False
```

## Answer: No!

**Slide 3**

- Counterexample:

```
data  Maybe a  where
    Nothing :: Maybe a
    Just    :: a -> Maybe a
```

```
l          = Cons "BLA" Nothing
eqM f _ _ = f l l
loop       = eqList eqM (==) l l
```

- We see that `loop` reduces to itself.

```
eqList eqM eq (Cons a mas) (Cons b mbs)
                      = eq a b
                        && eqM (eqList eqM eq) mas mbs
```

## Quiz Reloaded: Is `eqList` now terminating on all total inputs?

**Slide 4**

```
data  MList m a  where
    Nil  :: MList m a
    Cons :: a -> m (MList m a) -> MList m a
```

```
type Eq a = a -> a -> Bool
```

```
eqList :: (forall a. Eq a -> Eq (m a)) -> Eq a -> Eq (MList m a)
```

```
eqList eqM eq (Cons a mas) (Cons b mbs)
                      = eq a b
                        && eqM (eqList eqM eq) mas mbs
```

```
eqList ...
```

## Termination

**Slide 5**

- Question: *Will the run of a program eventually halt?*
- Undecidable for Turing-complete programming languages (Halteproblem).
- No termination checker can give a definitive answer for all programs.
- Problem still interesting for:
  - optimization and program specialization
  - total correctness of programs
  - theorem proving

## Termination for theorem proving

**Slide 6**

- Inductive theorem provers: e.g., Agda, Coq, Epigram, Twelf.
- Some proofs are *tree-shaped deriviations*, e.g., proof that $[a, 0] = [b, 0]$.

$$\dfrac{a = b \qquad \dfrac{0 = 0 \qquad [\,] = [\,]}{(0 :: [\,]) = (0 :: [\,])}}{a :: (0 :: [\,]) = b :: (0 :: [\,])}$$

- Some proofs are *recursive programs*, manipulating derivations.
- E.g., proof of $(l_1 = l_2) \to (l_2 = l_3) \to (l_1 = l_3)$.
- Only *terminating* programs denote valid proofs.
- E.g., program $\mathsf{let\ trans}\, d_1\, d_2 = \mathsf{trans}\, d_1\, d_2$ has to be rejected.

## Termination of Functions Over Inductive Types

**Slide 7**

- For termination, only structure of trees is interesting.
- Structure of these trees can be represented by *inductive types*.
- More inductive types:
  - lists
  - binary trees
  - natural numbers
  - tree ordinals

## Sized Inductive Types

**Slide 8**

- If $T$ is an inductive type, let $T^{\alpha}$ denote the set of its elements with *at most $\alpha$ constructors*.
- E.g., $\mathsf{List}^{\alpha}\ \mathsf{Int}$ contains integer lists of length $< \alpha$.
- $\mathsf{List}^{\omega}\ \mathsf{Int}$ is the type of all integer lists.
- In general, $T^{\infty}$ denotes the full type $T$.
- Sized list constructors:

$$\begin{array}{lcl} \mathsf{nil} & \in & \mathsf{List}^{\alpha+1}\,\mathsf{Int} \\ \mathsf{cons} & \in & \mathsf{Int} \to \mathsf{List}^{\alpha}\,\mathsf{Int} \to \mathsf{List}^{\alpha+1}\,\mathsf{Int} \end{array}$$

## A recursion principle from transfinite induction

**Slide 9**

- Rule for transfinite induction:

$$\frac{P(0) \qquad P(\alpha) \to P(\alpha+1) \qquad (\forall \alpha < \lambda.\, P(\alpha)) \to P(\lambda)}{P(\beta)}$$

- Recursive programs via fixed-point combinator $\mathsf{fix}\, f = f\,(\mathsf{fix}\, f)$.
- Instance $P(\alpha) = (\mathsf{fix}\, f \in A^{\alpha})$:
- Use transfinite induction to define a recursive program:

$$\frac{\mathsf{fix}\, f \in A^{0} \qquad f \in A^{\alpha} \to A^{\alpha+1} \qquad (\forall \alpha < \lambda.\, \mathsf{fix}\, f \in A^{\alpha}) \to \mathsf{fix}\, f \in A^{\lambda}}{\mathsf{fix}\, f \in A^{\beta}}$$

## Handling base and limit case

**Slide 10**

- Recursion principle:

$$\frac{\mathsf{fix}\, f \in A^{0} \qquad f \in A^{\alpha} \to A^{\alpha+1} \qquad (\mathsf{fix}\, f \in \bigcap_{\alpha<\lambda} A^{\alpha}) \to \mathsf{fix}\, f \in A^{\lambda}}{\mathsf{fix}\, f \in A^{\beta}}$$

- Restrict admissible types $A^{\alpha}$ such that
  - $\mathsf{fix}\, f \in A^{0}$ is trivial, e.g., $A^{\alpha} = T^{\alpha} \to C$,
  - $(\bigcap_{\alpha<\lambda} A^{\alpha}) \subseteq A^{\lambda}$.
- Specialized rule

$$\frac{\forall \alpha.\, f \in A^{\alpha} \to A^{\alpha+1}}{\mathsf{fix}\, f \in A^{\beta}}\, A^{\alpha} \text{ admissible}$$

## Type-Based Termination

- When termination checking a function clause

$$f : A^{\infty}$$
$$f\,p_1\,\ldots\,p_n = t(f),$$

**Slide 11**

- assume $f$ to be of type $A^{\alpha}$ on the right hand side,
- assume $f$ of type $A^{\alpha+1}$ on the left hand side,
- check well-typedness.
- For details and soundness, see draft of my thesis.

http://www.tcs.ifi.lmu.de/~abel/diss/

## Sized Monadic Lists

In context $\alpha : \mathsf{ord},\ M : * \xrightarrow{+} *,\ A : *$ we have

$$\begin{array}{lll} \mathsf{MList}^{\alpha}\,M\,A & : & * \\ \\ \mathsf{nil} & : & \mathsf{MList}^{\alpha+1}\,M\,A \\ \mathsf{cons} & : & A \to M\,(\mathsf{MList}^{\alpha}\,M\,A) \to \mathsf{MList}^{\alpha+1}\,M\,A \end{array}$$

**Slide 12**

## Solving the Quiz

**Slide 13**

- With $\mathsf{Eq}\,A = A \to A \to \mathsf{Bool}$ we can type monadic list equality as follows:

$$\mathsf{eqMList} : \forall M.\ (\forall A.\,\mathsf{Eq}\,A \to \mathsf{Eq}\,(M\ A)) \to \forall A.\,\mathsf{Eq}\,A \to \mathsf{Eq}\,(\mathsf{MList}^{\infty} M\ A)$$

$$\overbrace{\mathsf{eqMList}\ eqM\ \ eq}^{\mathsf{Eq}\,(\mathsf{MList}^{\alpha+1}\ M\ A)}\ \overbrace{(\mathsf{cons}\ a\ mas)}^{\mathsf{MList}^{\alpha+1}\ M\ A}\ (\mathsf{cons}\ b\ mbs) = eq\,a\,b\ \ \text{and}$$

$$\underbrace{eqM\ \overbrace{(\mathsf{eqMList}\ eqM\ \ eq)}^{\mathsf{Eq}\,(\mathsf{MList}^{\alpha}\ M\ A)}}_{\mathsf{Eq}\,M\,(\mathsf{MList}^{\alpha}\ M\ A)}\ \ \overbrace{mas}^{M\,(\mathsf{MList}^{\alpha}\ M\ A)}\ \ \ mbs$$

- A bit suprisingly, the quiz can be answered affirmatively.

## Related works on type-based termination

**Slide 14**

- Hughes, Pareto, Sabry (POPL 1996)
  *Proving the correctness of reactive systems using sized types*
- Amadio and Coupet-Grimal (FoSSaCS 1998)
  *Analysis of a guard condition in type theory*
- Xi (LICS 2001), *Prg. termination verification with dep. types*
- Chin, Khoo (HOSC 2001), *Calculating sized types*
- Barthe, Frade, Giménez, Pinto, Uustalu (MSCS 2004)
  *Type-based termination of recursive definitions*
- Blanqui (RTA 2004), *A type-based termination criterion for dependently-typed higher-order rewrite systems*
- Barthe et. al. (TLCA 2005): Inferring sized types
- Buchholz (2003), *Recursion on non-wellfounded trees*

## Acknowledgements

**Slide 15**

- Technical discussions on my thesis:

  Klaus Aehlig     Thorsten Altenkirch
  Martin Hofmann     John Hughes     Ralph Matthes
  Thomas Streicher     Tarmo Uustalu

- Stipends

  GKLI     CoVer

- Colleagues at Munich and Chalmers for support