Type-Based Termination for Dependent Types

Andreas Abel

INRIA, Team πr^2
PPS Lab, Paris

ProVal Seminar
INRIA Saclay, Parc Orsay Université, Paris, France
19 March 2010

Type-based termination

- View data (natural numbers, lists, binary trees) as trees.
- Type of data is equipped with a size.
- Size = upper bound on height of tree.
- Size must decrease in each recursive call.
- Termination is ensured by type-checker.

Sized types in a nutshell

- Sizes are upper bounds.
- List^a denotes lists of length $< a$.
- List^∞ denotes list of arbitrary (but finite) length.
- Sizes induce subtyping: $\mathrm{List}^a \leq \mathrm{List}^b$ if $a \leq b$.
- Size expressions a, b.

$$\begin{array}{rcll} a & ::= & i & \text{variable} \\ & | & a+1 & \text{successor} \\ & | & \infty & \omega \end{array}$$

Simple Example: Euclidean Division

```
data Nat : {i : Size} -> Set where
  zero : {i : Size} -> Nat {↑ i}
  suc  : {i : Size} -> Nat {i} -> Nat {↑ i}

sub : {i : Size} -> Nat {i} -> Nat {∞} -> Nat {i}
sub zero n = zero
sub (suc m) zero = suc m
sub (suc m) (suc n) = sub m n

-- div' m n  computes  ceiling(m/(n+1))
div' : {i : Size} -> Nat {i} -> Nat -> Nat {i}
div' zero    n = zero
div' (suc m) n = suc (div' (sub m n) n)
```

Termination Checking with Sized Types

```
sub : {i : Size} -> Nat {i} -> Nat {∞} -> Nat {i}
sub .{↑ i} (zero {i})  n        = zero {i}
sub .{↑ i} (suc {i} m) zero     = suc {i} m
sub .{↑ i} (suc {i} m) (suc n) = sub {i} m n

div' : {i : Size} -> Nat {i} -> Nat -> Nat {i}
div' .{↑ i} (zero {i})  n = zero {i}
div' .{↑ i} (suc {i} m) n =
  suc  {i} (div' {i} (sub {i} m n) n)
```

Note: you cannot match on sizes!

Internal handling of `Size`

- $\uparrow X$ unifies with ∞, yielding $X = \infty$.
- $i \leq \uparrow i \leq \infty$.
- A constraint solver for a set of inequalities $x + n \leq y, x \leq y + m$ (using Warshall's algorithm).
- Subtyping for sized data types:

$$\texttt{Nat}\ \{\texttt{i}\} \leq \texttt{Nat}\ \{\uparrow\ \texttt{i}\} \leq \texttt{Nat}\ \{\infty\}$$

- The strict inequality $i < \uparrow i$ is exploited for termination checking.

Example: Map for Rose Trees

```
data Rose (A : Set) : {_ : Size} -> Set where
  rose : {i : Size} -> A ->
         List (Rose A {i}) -> Rose A {↑ i}

mapRose : {A B : Set} -> (A -> B) ->
          {i : Size} -> Rose A {i} -> Rose B {i}
mapRose f .{↑ i} (rose {i} a l) = rose {↑ i} (f a)
  (map (mapRose f {i}) l)
```

Example: Map for General Trees

```
data Tree (A : Set)(B : Set) : {_ : Size} -> Set where
  tree : {i : Size} -> A ->
         (B -> Tree A {i}) -> Tree A {↑ i}

mapT : {A A' B : Set} -> (A -> A') ->
          {i : Size} -> Tree A B {i} -> Tree A' B {i}
mapT f .{↑ i} (tree {i} a t) = tree {↑ i} (f a)
  (λ b -> mapT f {i} (t b))
```

Formalization

- Sized inductive type $\mu^i X.\, A$.
- Equations and subtyping.

$$\begin{array}{lcll} \mu^{a+1} X.\, A & = & A[(\mu^a X.\, A)/X] & \\ \mu^{\infty} X.\, A & = & A[(\mu^{\infty} X.\, A)/X] & \\ \mu^a X.\, A & \leq & \mu^b X.\, A & \text{for } a \leq b \end{array}$$

- Example: lists.

$$\begin{array}{lcl} \mathsf{List}^i A & := & \mu^i X.\, 1 + A \times X \\ \\ \mathsf{nil} & : & \forall A \forall i.\, \mathsf{List}^{i+1} A \\ & := & \mathsf{inl}() \\ \\ \mathsf{cons} & : & \forall A.\, A \to \forall i.\, \mathsf{List}^i A \to \mathsf{List}^{i+1} A \\ & := & \lambda a \lambda as.\, \mathsf{inr}(a, as) \end{array}$$

Recursion

- Recursion principle (semantically):

$$\frac{\mathrm{fix}\, f \in A^0 \qquad f \in A^\alpha \to A^{\alpha+1} \qquad (\mathrm{fix}\, f \in \bigcap_{\alpha<\omega} A^\alpha) \to \mathrm{fix}\, f \in A^\omega}{\forall \beta \leq \omega.\, \mathrm{fix}\, f \in A^\beta}$$

- Step: $\mathrm{fix}\, f \in A^\alpha$ implies $f\,(\mathrm{fix}\, f) = \mathrm{fix}\, f \in A^{\alpha+1}$.
- Restrict admissible types A^α such that
  - $\mathrm{fix}\, f \in A^0$ is trivial, e.g., $A^\alpha = (\mu^\alpha X.A) \to C$, ($\mu^0 X.A$ is empty)
  - $(\bigcap_{\alpha<\omega} A^\alpha) \subseteq A^\omega$.
- Typing rule for recursion (e.g., $A^i = \mathrm{List}^i \mathrm{Int} \to \mathrm{List}^i \mathrm{Int}$):

$$\frac{f : \forall i.\, A^i \to A^{i+1}}{\mathrm{fix}\, f : A^a}\, A^i \text{ admissible}$$

Mutual Recursion and Measures

```
even  : {i : Size} -> Nat i -> Bool
even  n = even' n

even' : {i : Size} -> Nat i -> Bool
even' (zero)   = true
even' (succ n) = odd' n

odd'  : {i : Size} -> Nat i -> Bool
odd'  (zero)   = false
odd'  (succ n) = even n
```

Mutual Recursion and Measures

```
even  : {i : Size} -> {|i,↑0|} -> Nat i -> Bool
even {i} n = even' {i} n

even' : {i : Size} -> {|i,0|} -> Nat i -> Bool
even' .{↑ i} (zero {i})   = true
even' .{↑ i} (succ {i} n) = odd' {i} n

odd'  : {i : Size} -> {|i,0|} -> Nat i -> Bool
odd'  .{↑ i} (zero {i})   = false
odd'  .{↑ i} (succ {i} n) = even {i} n
```

Mutual Recursion and Measures

```
even  : {i : Size} -> {|i,↑0|} -> Nat i -> Bool
even {i} {|i,↑0|} n = even' {i} {|i,0|} n

even' : {i : Size} -> {|i,0|} -> Nat i -> Bool
even' .{↑ i} {|↑i,0|} (zero {i})   = true
even' .{↑ i} {|↑i,0|} (succ {i} n) = odd' {i} {|i,0|} n

odd'  : {i : Size} -> {|i,0|} -> Nat i -> Bool
odd'  .{↑ i} {|↑i,0|} (zero {i})   = false
odd'  .{↑ i} {|↑i,0|} (succ {i} n) = even {i} {|i,↑0|} n
```

Typing rules for measures

- Introduction

$$\frac{\Gamma \vdash_\mu t : T}{\Gamma \vdash \lambda\_t : \mu \to T}$$

- Elimination

$$\frac{\Gamma \vdash_\mu t : \mu' \to T \qquad \mu' < \mu}{\Gamma \vdash t\ \_ : T}$$

PTS with Size

- Axioms: $(\mathsf{Prop}, \mathsf{Set}), (\mathsf{Set}, \mathsf{Type}), (\mathsf{Size}, \mathsf{Type})$.
- New rules: (Size, s, s).
- No rules (s, Size, s') except $(\mathsf{Size}, \mathsf{Size}, \mathsf{Size})$.

Conclusions

- MiniAgda has η, $\forall$ and sized types.
- Add type-based termination with invisible $\forall i : \mathsf{Size}$ to Coq!