Some theory about nothing

Nils Anders Danielsson

Division meeting, Aspenäs, 2019-09-20

`@0`

- `@0` is used to mark arguments and definitions that should be erased at run-time.
- Agda is supposed to make sure that:
  - Things marked as erased are actually erased.
  - There is never any data missing at run-time.
- The typing rules are based on work by McBride and Atkey.
- Andreas is working on the implementation.

@0

```
ok : {@0 A : Set} → A → A
ok x = x

-- not-ok : {@0 A : Set} → @0 A → A
-- not-ok x = x

-- Not-ok : @0 Bool → Set
-- Not-ok true  = ⊤
-- Not-ok false = ⊥
```

Erased

A type-level variant of @0:

```
record Erased (@0 A : Set a) : Set a where
  constructor [_]
  field
    @0 erased : A

open Erased public
```

Monad

Erased is a monad:

$$\mathsf{return} : \{@0\ A : \mathsf{Set}\ a\} \to @0\ A \to \mathsf{Erased}\ A$$
$$\mathsf{return}\ x = [\ x\ ]$$

$$\_{\gg}\!=\_ :$$
$$\quad \{@0\ A : \mathsf{Set}\ a\}\ \{@0\ B : \mathsf{Set}\ b\} \to$$
$$\quad \mathsf{Erased}\ A \to (A \to \mathsf{Erased}\ B) \to \mathsf{Erased}\ B$$
$$x \gg\!= f = [\ \mathsf{erased}\ (f\ (\mathsf{erased}\ x))\ ]$$

An application

An application

I have tried to define natural numbers that compute (roughly) like unary natural numbers at compile-time, but like binary natural numbers at run-time.

The underlying representation

Binary natural numbers:

$$\text{Bin'} : \text{Set}$$
$$\text{Bin'} = \text{List Bool}$$

The representation of a given natural number is not unique. A split surjection:

$$\text{to-}\mathbb{N} : \text{Bin'} \to \mathbb{N}$$

Indexed binary numbers

Binary natural numbers representing a given natural number:

```
abstract

  Bin-[_] : @0 ℕ → Set
  Bin-[ n ] =
    ∥ (Σ Bin′ λ b → Erased (to-ℕ b ≡ n)) ∥
```

- Abstract so the underlying representation can be changed without breaking client code.
- Truncated so that the representation is unique.

Non-indexed binary numbers

Binary natural numbers:

$$\mathsf{Bin} : \mathsf{Set}$$
$$\mathsf{Bin} = \Sigma\ (\mathsf{Erased}\ \mathbb{N})\ \lambda\ n \to \mathsf{Bin}\text{-}[\ \mathsf{erased}\ n\ ]$$

Returns the erased index:

$$@0\ \lfloor\_\rfloor : \mathsf{Bin} \to \mathbb{N}$$
$$\lfloor\ [\ n\ ]\ ,\ \_\ \rfloor = n$$

[]-cong

A key lemma:

```
[]-cong :
  {@0 A : Set a} {@0 x y : A} →
  Erased (x ≡ y) → [ x ] ≡ [ y ]
```

[]-cong

A key lemma:

$$\begin{array}{l} \mathrm{[]\text{-}cong} : \\ \quad \{@0\ A : \mathsf{Set}\ a\}\ \{@0\ x\ y : A\} \rightarrow \\ \quad \mathsf{Erased}\ (x \equiv y) \rightarrow [\ x\ ] \equiv [\ y\ ] \end{array}$$

With the K rule and propositional equality:

$$\mathrm{[]\text{-}cong}\ [\ \mathsf{refl}\ ] = \mathsf{refl}$$

[]-cong

A key lemma:

$$
\begin{array}{l}
[]\text{-cong} : \\
\quad \{@0\ A : \mathsf{Set}\ a\}\ \{@0\ x\ y : A\} \rightarrow \\
\quad \mathsf{Erased}\ (x \equiv y) \rightarrow [\ x\ ] \equiv [\ y\ ]
\end{array}
$$

With the K rule and propositional equality:

$$[]\text{-cong}\ [\ \mathsf{refl}\ ] = \mathsf{refl}$$

With Cubical Agda and paths:

$$[]\text{-cong}\ [\ eq\ ] = \lambda\ i \rightarrow [\ eq\ i\ ]$$

[]-cong

A key lemma:

$$
\begin{array}{l}
[]\text{-cong} : \\
\quad \{@0\ A : \mathsf{Set}\ a\}\ \{@0\ x\ y : A\} \to \\
\quad \mathsf{Erased}\ (x \equiv y) \to [\ x\ ] \equiv [\ y\ ]
\end{array}
$$

With the K rule and propositional equality:

$$[]\text{-cong}\ [\ \mathsf{refl}\ ] = \mathsf{refl}$$

With Cubical Agda and paths:

$$[]\text{-cong}\ [\ eq\ ] = \lambda\ i \to [\ eq\ i\ ]$$

In both cases $[]\text{-cong}$ is an equivalence that maps $[\ \mathsf{refl}\ x\ ]$ to $\mathsf{refl}\ [\ x\ ]$.

Non-indexed binary numbers

Recall:

$$\mathsf{Bin} : \mathsf{Set}$$
$$\mathsf{Bin} = \Sigma\ (\mathsf{Erased}\ \mathbb{N})\ \lambda\ n \to \mathsf{Bin\text{-}[}\ \mathsf{erased}\ n\ \mathsf{]}$$

$$@0\ \lfloor\_\rfloor : \mathsf{Bin} \to \mathbb{N}$$
$$\lfloor\ [\ n\ ]\ ,\ \_\ \rfloor = n$$

Equality follows from equality for the erased indices:

$$\mathsf{Erased}\ (\lfloor\ x\ \rfloor \equiv \lfloor\ y\ \rfloor) \simeq (x \equiv y)$$

Addition

```
abstract

  plus : {@0 m n : ℕ} →
         Bin-[ m ] → Bin-[ n ] → Bin-[ m + n ]
  plus = ... -- Add with carry.

_⊕_ : Bin → Bin → Bin
([ m ] , x) ⊕ ([ n ] , y) = [ m + n ] , plus x y
```

Conversion to/from unary natural numbers?

Goal:

- $\mathsf{Bin} \simeq \mathbb{N}$ (in a non-erased context).
- With the forward direction pointwise equal to $\lfloor\_\rfloor$ (in an erased context).

Stability

Stability

A type A is *stable* if Erased A implies A:

$$\text{Stable} : \text{Set } a \to \text{Set } a$$
$$\text{Stable } A = \text{Erased } A \to A$$

A type is *very stable* if [_] is an equivalence:

$$\text{Very-stable} : \text{Set } a \to \text{Set } a$$
$$\text{Very-stable } A = \text{Is-equivalence } ([\_]\ \{A = A\})$$

Double negation

Erased A implies $\neg \neg A$. Thus types that are stable for double negation are stable for Erased:

$$\{@0\ A : \mathsf{Set}\ a\} \to (\neg \neg A \to A) \to \mathsf{Stable}\ A$$

Types for which it is known whether or not they are inhabited are also stable:

$$\{@0\ A : \mathsf{Set}\ a\} \to A \uplus \neg A \to \mathsf{Stable}\ A$$

Stability of equality

Variants of Stable and Very-stable:

$$\text{Stable-}{\equiv} : \text{Set } a \to \text{Set } a$$
$$\text{Stable-}{\equiv}\ A = \{x\ y : A\} \to \text{Stable } (x \equiv y)$$

$$\text{Very-stable-}{\equiv} : \text{Set } a \to \text{Set } a$$
$$\text{Very-stable-}{\equiv}\ A = \{x\ y : A\} \to \text{Very-stable } (x \equiv y)$$

Decidable equality

Stable propositions are very stable:

$$\text{Stable } A \to \text{Is-proposition } A \to \text{Very-stable } A$$

Thus types for which equality is decidable have very stable equality:

$$((x\ y : A) \to x \equiv y \uplus \neg\, x \equiv y) \to \text{Very-stable-}{\equiv}\ A$$

Propositions

However, it is not the case that every very stable type is a proposition:

$$\neg\ (\{A : \mathsf{Set}\ a\} \to \mathsf{Very\text{-}stable}\ A \to \mathsf{Is\text{-}proposition}\ A)$$

Erased Bool is not a proposition, but it is very stable:

$$\{@0\ A : \mathsf{Set}\ a\} \to \mathsf{Very\text{-}stable}\ (\mathsf{Erased}\ A)$$

Closure properties

Closure properties for Stable, Very-stable, Stable-$\equiv$ and Very-stable-$\equiv$.

Back to the application

An equivalence

A lemma:

$$\{@0\ y : A\} \to$$
$$\text{Very-stable-}{\equiv}\ A \to$$
$$\text{Is-proposition}\ (\Sigma\ A\ \lambda\ x \to \text{Erased}\ (x \equiv y))$$

This lemma is used below (where n is erased):

$$
\begin{aligned}
&\text{Bin-[}\ n\ \text{]} && \simeq \\
&\|\ (\Sigma\ \text{Bin'}\ \lambda\ b \to \text{Erased}\ (\text{to-}\mathbb{N}\ b \equiv n))\ \| && \simeq \\
&\|\ (\Sigma\ \mathbb{N}\ \lambda\ m \to \text{Erased}\ (m \equiv n))\ \| && \simeq \\
&(\Sigma\ \mathbb{N}\ \lambda\ m \to \text{Erased}\ (m \equiv n))
\end{aligned}
$$

Another equivalence

Finally we can prove that the binary natural numbers are equivalent to the unary ones:

$$
\begin{array}{ll}
\mathsf{Bin} & \simeq \\
(\Sigma\ (\mathsf{Erased}\ \mathbb{N})\ \lambda\ n \to \mathsf{Bin\text{-}[}\ \mathsf{erased}\ n\ \mathsf{]}) & \simeq \\
(\Sigma\ (\mathsf{Erased}\ \mathbb{N})\ \lambda\ n \to \Sigma\ \mathbb{N}\ \lambda\ m \to & \\
\quad \mathsf{Erased}\ (m \equiv \mathsf{erased}\ n)) & \simeq \\
(\Sigma\ \mathbb{N}\ \lambda\ m \to \Sigma\ (\mathsf{Erased}\ \mathbb{N})\ \lambda\ n \to & \\
\quad \mathsf{Erased}\ (m \equiv \mathsf{erased}\ n)) & \simeq \\
(\Sigma\ \mathbb{N}\ \lambda\ m \to \mathsf{Erased}\ (\Sigma\ \mathbb{N}\ \lambda\ n \to m \equiv n)) & \simeq \\
\mathbb{N} \times \mathsf{Erased}\ \top & \simeq \\
\mathbb{N} \times \top & \simeq \\
\mathbb{N} &
\end{array}
$$

Another equivalence

Finally we can prove that the binary natural numbers are equivalent to the unary ones:

$$\mathsf{Bin} \simeq \mathbb{N}$$

In an erased context the forward direction is pointwise equal to $\lfloor \_ \rfloor$ (i.e. it returns the index).

Discussion

- There is currently no compiler for Cubical Agda, so the run-time performance of the binary numbers has not been tested.
- I have also used the same technique to implement a FIFO queue transformer:
  - The enqueue function computes (roughly) like the corresponding list function, but not dequeue.
  - The dequeue function requires that equality is very stable for the carrier type.

Discussion

- A surprising amount of theory for something as simple as Erased?

Some theory

Some equivalences

Easy to prove:

$$\begin{aligned}
&\text{Erased } \bot \simeq \bot \\
&\text{Erased } \top \simeq \top \\
&\text{Erased } ((x : A) \to P\ x) \simeq ((x : A) \to \text{Erased } (P\ x)) \\
&\text{Erased } (\Sigma\ A\ P) \simeq \\
&\quad \Sigma\ (\text{Erased } A)\ (\lambda\ x \to \text{Erased } (P\ (\text{erased } x)))
\end{aligned}$$

If equality is extensional and the pattern [sup x f] is OK:

$$\begin{aligned}
&\text{Erased } (\text{W } A\ P) \simeq \\
&\quad \text{W } (\text{Erased } A)\ (\lambda\ x \to \text{Erased } (P\ (\text{erased } x)))
\end{aligned}$$

Some preservation lemmas

For erased A : Set a and B : Set b:

$$
\begin{array}{l}
@0\ (A \to B) \to \text{Erased}\ A \to \text{Erased}\ B \\
@0\ A \Leftrightarrow B \to \text{Erased}\ A \Leftrightarrow \text{Erased}\ B \\
@0\ A \twoheadrightarrow B \to \text{Erased}\ A \twoheadrightarrow \text{Erased}\ B \\
@0\ A \leftrightarrow B \to \text{Erased}\ A \leftrightarrow \text{Erased}\ B \\
@0\ A \simeq B \to \text{Erased}\ A \simeq \text{Erased}\ B \\
@0\ A \rightarrowtail B \to \text{Erased}\ A \rightarrowtail \text{Erased}\ B \\
@0\ \text{Embedding}\ A\ B \to \\
\quad \text{Embedding}\ (\text{Erased}\ A)\ (\text{Erased}\ B)
\end{array}
$$

H-levels

Erased commutes with H-level n:

$$\text{Erased}\ (\text{H-level}\ n\ A) \Leftrightarrow \text{H-level}\ n\ (\text{Erased}\ A)$$

Closure properties

Closure properties

For Stable:

$$\mathsf{Stable}\ \bot$$
$$\mathsf{Stable}\ \top$$
$$(\forall\ x \to \mathsf{Stable}\ (P\ x)) \to \mathsf{Stable}\ ((x : A) \to P\ x)$$

For Very-stable and Stable:

$$\begin{aligned}&\mathsf{Very\text{-}stable}\ A \to (\forall\ x \to \mathsf{Stable}\ (P\ x)) \to \\ &\quad \mathsf{Stable}\ (\Sigma\ A\ P)\end{aligned}$$

Closure properties

For Very-stable (in some cases assuming that equality is extensional):

$$
\begin{array}{l}
\text{Very-stable } \bot \\
\text{Very-stable } \top \\
(\forall\ x \to \text{Very-stable } (P\ x)) \to \\
\quad \text{Very-stable } ((x : A) \to P\ x) \\
\text{Very-stable } A \to (\forall\ x \to \text{Very-stable } (P\ x)) \to \\
\quad \text{Very-stable } (\Sigma\ A\ P) \\
\text{Very-stable } A \to \text{Very-stable } (\mathrm{W}\ A\ P)
\end{array}
$$

Closure properties

If A is very stable, then equality is very stable for A:

$$\text{Very-stable } A \to \text{Very-stable-}{\equiv}\ A$$

Closure properties

For Stable-$\equiv$ (in one case assuming that equality is extensional):

$$
\begin{array}{l}
\text{Stable-}\!\equiv A \to \text{Stable-}\!\equiv B \to \text{Stable-}\!\equiv (A \uplus B) \\
(\forall\ x \to \text{Stable-}\!\equiv (P\ x)) \to \text{Stable-}\!\equiv ((x :\ A) \to P\ x) \\
\text{Stable-}\!\equiv A \to \text{Stable-}\!\equiv (\text{List}\ A)
\end{array}
$$

For Very-stable-$\equiv$ and Stable-$\equiv$:

$$
\begin{array}{l}
\text{Very-stable-}\!\equiv A \to (\forall\ x \to \text{Stable-}\!\equiv (P\ x)) \to \\
\quad \text{Stable-}\!\equiv (\Sigma\ A\ P)
\end{array}
$$

Closure properties

For Very-stable-$\equiv$ (in some cases assuming that equality is extensional):

$$
\begin{array}{l}
\text{Very-stable-}{\equiv}\ A \to \text{Very-stable-}{\equiv}\ B \to \\
\quad \text{Very-stable-}{\equiv}\ (A \uplus B) \\
(\forall\ x \to \text{Very-stable-}{\equiv}\ (P\ x)) \to \\
\quad \text{Very-stable-}{\equiv}\ ((x : A) \to P\ x) \\
\text{Very-stable-}{\equiv}\ A \to (\forall\ x \to \text{Very-stable-}{\equiv}\ (P\ x)) \to \\
\quad \text{Very-stable-}{\equiv}\ (\Sigma\ A\ P) \\
\text{Very-stable-}{\equiv}\ A \to \text{Very-stable-}{\equiv}\ (\mathrm{W}\ A\ P) \\
\text{Very-stable-}{\equiv}\ A \to \text{Very-stable-}{\equiv}\ (\mathrm{List}\ A)
\end{array}
$$