Landau in Agda

Talk in Bath, 2009.10.22

CVS, AIST
Yoshiki Kinoshita

Contents

- Use of formal proofs
- Landau's "Grundlagen der Analysis"
- A tool for presenting equational reasoning
- Comparison of Landau's original proof and proof suitable for Agda

Use of formal proofs

- To give evidences of theorems.
  For this purpose, "readability" is not the issue.
- To understand the theorem
  side-effect?
  Action of proving a theorem (may) help understand it better.

Some mathematicians claim they know its truth before they prove it.

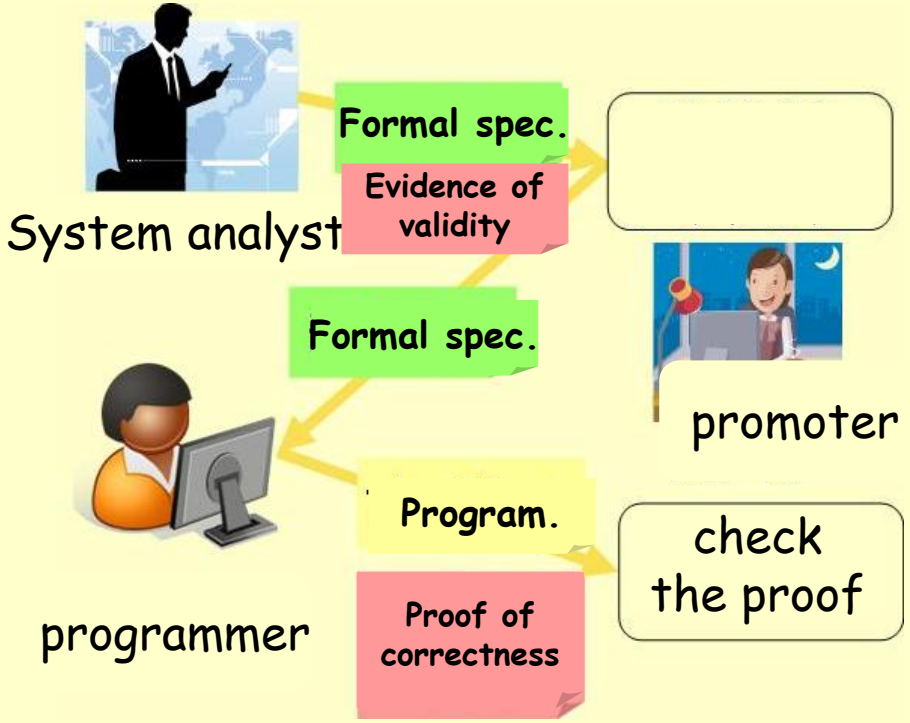
Proving and understanding are different action.

Writing a *formal* proof does not help understanding the theorem (?)

Use of formal proofs in system construction

Formal proofs may be used as an **evidence** that the system (software product) is not a junk.

For that, readability is not very much an issue.



Checking formal proofs

Propositions to be proven in programming tends

- to be simple and not to be hard,
  compared to many propositions to be shown in mathematics
- but to have many cases to analyse.

Cf. treatment of algebraic structures in math. and CS:

So, checking proofs is useful.

Readability of proofs

- Agda emphasises human readable formal proofs.
- With an appropriate abstraction (which may be variable according to the context), formal proofs should be read by human being.
- Today's talk: notation for equational-inequational reasoning.

Writing Landau in Agda

- I had to do some arithmetic in Agda
- There is a standard library for natural number arithmetic in Agda,
  - But to me it looked easier to write my own library than to understand a sophisticated code in the std lib.
- So, use Landau as the list of useful propositions for a library.
  - Nice intro. to Landau written by Teiji Takagi in 1935 (Landau's book published in 1930).
  - Target of Automath!
- Currently, only addition has been done.

Peano's axioms in Agda

Landau's natural number starts with 1,

I make it start with 0, to follow recent trend (and make (N, +, 0) a monoid).

```
data ℕ : Set where
  zero : ℕ
    {- Axiom1 : 0 : ℕ -}
  suc : ℕ → ℕ
    {- Axiom2 : x : ℕ → suc x :
    ℕ -}

Axiom3 : ∀ {x} → suc x ≢ 0
Axiom3 {0} ()
Axiom3 {suc x} ()
```

Peano's axioms in Agda

```
Axiom4 :
  ∀ {x y} → suc x ≡ suc y → x ≡ y
Axiom4 {0} .{0} refl = refl
Axiom4 {suc x} .{suc x} refl = refl

Axiom5 :
  ∀ (P : ℕ → Set) →
    P 0 →
    (∀ (m : ℕ) → P m → P (suc m)) →
    ∀ (n : ℕ) → P n
Axiom5 P p0 _ 0 = p0
Axiom5 P p0 ps (suc n) = ps n
     (Axiom5 P p0 ps n)
```

A syntax sugar for induction

$$\mathbb{N}\text{-Induction\_Base\_Step\_} :$$
$$\forall\ (P : \mathbb{N} \to \mathrm{Set}) \to$$
$$P\ 0 \to$$
$$(\forall\ (m : \mathbb{N}) \to P\ m \to P\ (\mathrm{suc}\ m)) \to$$
$$\forall\ (n : \mathbb{N}) \to P\ n$$
$$\mathbb{N}\text{-Induction}\ P\ \mathrm{Base}\ p0\ \mathrm{Step}\ ps =$$
$$\mathrm{Axiom5}\ P\ p0\ ps$$

Order

In Agda: introducion of the order before the addition.

to introduce and use the Equational-Inequational reasoning notation for proving theorems about addition.

Our notation for Equational-Inequational reasoning:

p : a < z

p = ∵ a ≡ b by prop1
≤ c by prop2
< d by prop3
≡ z by prop4 ■

```
data _≤_ : ℕ → ℕ → Set where
  z≤n : ∀ {n} → 0 ≤ n
  s≤s : ∀ {m n} →
    (m≤n : m ≤ n) → suc m ≤ suc n
```

```
_<_ : ℕ → ℕ → Set
m < n = suc m ≤ n
_≥_ : ℕ → ℕ → Set
m ≥ n = n ≤ m
_>_ : ℕ → ℕ → Set
m > n = n < m
```

> vs ≤

Landau:

- $x > y$ iff $x = y + u$, i.e., $(\exists\ u)\ x = y + u$
- $>$ rather than $\geq$ because numbers starts with 1.

Agda:

- $\leq$ rather than $<$ because numbers starts with 0.
- Inductive definition rather than definition using $\exists$;
  - tendency to avoid $\exists$ and $\vee$ in Agda.

Some propositions.

Note the freedom in choice of characters in identifiers in Agda.

```
x≤sy→x≤y : ∀ {x y} →
  suc x ≤ suc y → x ≤ y
sx≤sy→x≤y (s≤s p) = p

s≡s : ∀ {x y} →
  x ≡ y → suc x ≡ suc y
s≡s {x} {y} =
   Congruence {ℕ} {ℕ}
   {x} {y} suc
```

Propositions of the form k ~ m → m @ n → k # n

Propositions of the form

k ~ m → m @ n → k # n

with ~ and @ replaced by one of ≡, ≤ or < (i.e., nine propositions) are necessary to provide our notation for Equational-Inequational reasoning

See Theorems 15-17

≡-≡ : ∀ {k m n} → k ≡ m → m ≡ n → k ≡ n
≡-≤ : ∀ {k m n} → k ≡ m → m ≤ n → k ≤ n
≡-< : ∀ {k m n} → k ≡ m → m < n → k < n
≤-≡ : ∀ {k m n} → k ≤ m → m ≡ n → k ≤ n
≤-≤ : ∀ {k m n} → k ≤ m → m ≤ n → k ≤ n
≤-< : ∀ {k m n} → k ≤ m → m < n → k < n
<-≡ : ∀ {k m n} → k < m → m ≡ n → k < n
<-≤ : ∀ {k m n} → k < m → m ≤ n → k < n
<-< : ∀ {k m n} → k < m → m < n → k < n

Notation for equational-inequational reasoning

```
p : a < z
p = ∵ a ≡ b  by prop1
      ≤ c  by prop2
      < d  by prop3 ∎
```

```
infix 1  _∎
infixl 2 _≡_by_ _≤_by_ _<_by_
infix 3 ∵_

data _~_ (x y : ℕ) : Set where
  eq : (x≡y : x ≡ y) → x ~ y
  lt : (x<y : x < y) → x ~ y
  le : (x≤y : x ≤ y) → x ~ y
```

```
_∎ : ∀ {x y} → (r : x ~ y) → ∎-definition.C r
eq p ∎ = p
le p ∎ = p
lt p ∎ = p

∵_ : ∀ (a : ℕ) → a ~ a
∵_ a = eq refl

_≡_by_ : ∀ {x y} →
  x ~ y → (z : ℕ) → y ≡ z → x ~ z
eq p ≡ _ by r = eq (≡-≡ p r)
le p ≡ _ by r = le (≤-≡ p r)
lt p ≡ _ by r = lt (<-≡ p r)

_≤_by_ : ∀ {x y} →
  x ~ y → (z : ℕ) → y ≤ z → x ~ z
_<_by_ : ∀ {x y} →
  x ~ y → (z : ℕ) → y < z → x ~ z
```

Landau:

- Theorems tends to be in negation
  - Cf. Theorem1, the contraposition of Axiom4.)
- Why? Use of Reductio ad absurdum?

Agda :

- Negation not preferred.
- Axiom4 would do.

```
Theorem1 : ∀ {x y} →
  x ≢ y → suc x ≢ suc y
Theorem1 p q = p (Axiom4
    q)
```

Avoiding V

- Definitions

  instead of merging two definitions by V,
  provide a definition anew.

- Theorems of sequent-like form:

  $A \rightarrow B \rightarrow \ldots \rightarrow C \rightarrow (D \vee E \vee \ldots \vee F)$

  where A, B, …, F are atomic formulae or their negations.

Definition of $\leqq$ and $\geqq$

Landau's definition:

$x \leqq y$ iff $x < y \vee x = y$

which boils the argument on $\leqq$ down to that on $<$ and $=$.

In Agda, however, one tends to give individual inductive definition to $\leqq$ and then prove the above as a theorem.

$\vee$-elimination vs. case analysis.

Theorem10 : ∀ (x y : ℕ) → x ≡ y ∨ x > y ∨ x < y

```
Theorem10 0 0 = inl (refl)
Theorem10 0 (suc y) = inr
   (inr (s≤s z≤n))
Theorem10 (suc x) 0 = inr
   (inl (s≤s z≤n))
```

```
Theorem10 (suc x) (suc y)
  =
 ∨-elim3
  (Theorem10 x y)
  (λ x≡y →
    inl (s≡s x≡y))
  (λ suc[x]≤y →
    inr (inl (s≤s suc[x]≤y)))
  (λ suc[y]≤x →
    inr (inr (s≤s suc[y]≤x)))
```

These are all variable names!

In Agda, Theorems 15-17 has been proved with the introduction of order.

Landau did not consider mixture of equations and inequations, so there are only four cases.

```
Theorem15 : ∀ {x y z : ℕ} →
    x < y → y < z → x < z
Theorem15 = <-<
Theorem16-1 : ∀ {x y z : ℕ}
    → x ≤ y → y < z → x < z
Theorem16-1 = ≤-<
Theorem16-2 : ∀ {x y z : ℕ}
    → x < y → y ≤ z → x < z
Theorem16-2 = <-≤
Theorem17 : ∀ {x y z : ℕ} →
    x ≤ y → y ≤ z → x ≤ z
Theorem17 = ≤-≤
```

Presentation form of theorems

Landau's Theorem19 could be divided into three, for more convenient use, as I did in Agda.

Similar holds for Theorem20:

Theorem20 : ∀ (x y z : ℕ) →
(x + z > y + z → x > y)
∧ (x + z ≡ y + z → x ≡ y)
∧ (x + z < y + z → x < y)

Theorem19 : ∀ (x y z : ℕ)
→ (x > y → x + z > y + z)
∧ (x ≡ y → x + z ≡ y + z)
∧ (x < y → x + z < y + z)

Theorem19-1 : ∀ (x y z : ℕ) →
x > y → x + z > y + z
Theorem19-2 : ∀ (x y z : ℕ) →
x ≡ y → x + z ≡ y + z
Theorem19-3 : ∀ (x y z : ℕ) →
x < y → x + z < y + z

Currying

Theorem21 is in good shape: only we change $A \wedge B \to C$ to $A \to B \to C$ (currying!).

```
Theorem21 : ∀ (x y z u : ℕ) →
  x > y → z > u → x + z > y + u
```

$A \vee B \rightarrow C \Leftrightarrow (A \rightarrow C) \wedge (B \rightarrow C)$

Theorem22:

$$\forall (x\ y\ z\ u : \mathbb{N}) \rightarrow$$
$$x \geq y \ \wedge \ z > u \ \ \vee \ \ x > y \ \wedge \ z \geq u \rightarrow$$
$$x + z > y + u$$

more conveniently formulated as two theorems:

$$\forall (x\ y\ z\ u : \mathbb{N}) \rightarrow$$
$$x \geq y \ \wedge \ z > u \ \ \rightarrow \ \ x + z > y + u$$

$$\forall (x\ y\ z\ u : \mathbb{N}) \rightarrow$$
$$x > y \ \wedge \ z \geq u \ \ \rightarrow \ \ x + z > y + u$$

Observations I

- There is a logically equivalent form which is suited to Agda (and other provers).
  - Try to make the theorem in sequent-like form:

    $A \to B \to \ldots \to C \to (D \lor E \lor \ldots \lor F)$

    where A, B, …, F are atomic formulae or their negations.
    - Presentation of theorems ($A \lor B \to C$ or two theorems: $A \to C$ and $B \to C$)
    - $A \to B \to C \to D$ rather than $A \land B \land C \to D$
  - Cf. def. of order. I used inductive definition instead of Landau's definition by means of ∃.

Observations II

- Introduction of our notation for Equational-Inequational reasoning increases the readability.
  - OBJECTION: such equations and inequations which are decidable should be hidden and automatically checked by the computer. Cf. Coq-MT

    The notation is still useful for undecidable relations.

    The notation could be used even for decidable relations to make the *human reader* of the proof understand the proof easier.

Observations III

- Wider choice of characters for identifers helps a lot.

Fin