Inductive Definitions and Type Theory an Introduction

Thierry Coquand

Preliminary draft for the TYPES Summer School, August 1999

1 A Normalisation Proof of Gödel's System T

Gödel system T is one of the simplest type system with one inductive definition. In this section, we present a proof of normalisation for *head-reduction* and closed terms. This proof uses as a main tool one inductive definition, of being *reducible* or *computable* at type N.

In order to present a *complete* proof (in particular, to be completely precise about *bound* variables) we use a simple form of *explicit substitution*. The proof is formulated in such a way that it follows the following computational interpretation: an object u of type N is a (well-founded) method that will eventually gives 0 or $S\ u_1$ with u_1 method of type N. A method of type $\alpha \to \beta$ is a method that applied to a method of type α with gives a method of type β.

The goal of the argument is to prove that all closed terms are correct methods in this sense. For doing this, we have to explain what should be the meaning of an *open* term as well: it should correspond to an *hypothetical* method. Explicit substitutions are also used to explain this notion.

1.1 Gödel's system T

The types are inductively defined as being N or of the form $\alpha \to \beta$. A *context* $\Gamma, \Delta, \ldots$ is a set of typed variables $x_1 : \alpha_1, \ldots, x_n : \alpha_n$ with $x_1, \ldots, x_n$ distinct. The typing rules are the following

- $\Gamma \vdash x : \alpha$ if $x : \alpha$ is in Γ,
- $\Gamma \vdash 0 : N$,
- $\Gamma \vdash S\ t : N$ if $\Gamma \vdash t : N$,
- $\Gamma \vdash t\ t' : \beta$ if $\Gamma \vdash t : \alpha \to \beta$ and $\Gamma \vdash t' : \alpha$,
- $\Gamma \vdash \lambda x\ t : \alpha \to \beta$ if x does not appear in Γ and $\Gamma, x : \alpha \vdash t : \beta$,
- $\Gamma \vdash f_{a,b} : N \to \alpha$ if $\Gamma \vdash a : \alpha$ and $\Gamma \vdash b : N \to \alpha \to \alpha$.

We recall that $\alpha \to \alpha_1 \to \alpha_2$ should be read $\alpha \to (\alpha_1 \to \alpha_2)$. If we have $\Gamma \vdash t : \alpha$ then t represents an *hypothetical* method of type α with free variables Γ. If $\Gamma = x_1 : \alpha_1, \ldots, x_n : \alpha_n$ and u_i is a closed expression of type α_i then we say that the substitution $\gamma = (x_1 = u_1, \ldots, x_n = u_n)$ is a (closed) instantiation of Γ. We can then form the instantiation $t\gamma$ of the method t and we get a closed expression of type α. Recursively, the closed expressions are hence described by the following clauses:

- 0 is a closed expression of type N,

- $S\ u$ is a closed expression of type N, if u is a closed expression of type N,
- $u\ u'$ is a closed expression of type β, if u, u' are closed expressions of type $\alpha \to \beta,\ \alpha$ respectively,
- $t\gamma$ is a closed expression of type α if $\Gamma \vdash t : \alpha$ and γ is an instantiation of Γ.

We describe next the head-reduction on closed expressions:

- $0\gamma \to 0$,
- $(S\ t)\gamma \to S\ (t\gamma)$,
- $(\lambda x\ t)\gamma\ u \to t(\gamma, x = u)$,
- $u\ u' \to u_1\ u'$ if $u \to u_1$,
- $f_{a,b}\gamma\ 0 \to a\gamma$,
- $f_{a,b}\gamma\ (S\ u) \to b\gamma\ u\ (f_{a,b}\gamma\ u)$,
- $f_{a,b}\gamma\ u \to f_{a,b}\gamma\ u_1$ if $u \to u_1$.

As usual, let us write $\to^*$ for the reflexive transitive closure of $\to$. We define inductively the predicate $R_N\ u$ which expresses that the closed expression u is *reducible* or *computable* of type N:

$$\frac{u \to^* 0}{R_N\ u} \qquad\qquad \frac{u \to^* S\ u_1 \qquad R_N\ u_1}{R_N\ u}$$

and we define recursively on the type α what it means to be reducible at any type: $R_{\alpha\to\beta}\ u$ means that $R_\alpha\ u'$ implies $R_\beta\ (u\ u')$.

1.2 All terms are computable

The proofs is divided in two lemmas.

Lemma 1: If $R_\alpha\ u'$ and $u \to u'$ then $R_\alpha\ u$.

Corollary: If $R_\alpha\ u'$ and $u \to^* u'$ then $R_\alpha\ u$.

Lemma 2: If $x_1 : \alpha_1, \ldots, x_n : \alpha_n \vdash t : \alpha$ and $R_{\alpha_i}\ u_i$ then we have $R_\alpha\ t(x_1 = u_1, \ldots, x_n = u_n)$.

Theorem: All closed expressions are reducible.

Let us prove the theorem, assuming first lemma 2. We do an induction on the form of a closed expression v:

- $v = 0$ is reducible by definition,
- $v = S\ u$, by induction hypothesis, u is reducible, hence so is v,
- $v = u\ u'$ by induction hypothesis, both u and u' are reducible, hence so is v,
- $v = t(x_1 = u_1, \ldots, x_n = u_n)$, by induction hypothesis all $u_1, \ldots, u_n$ are reducible, hence, by lemma 2, so is v.

Hence the theorem is easily proved if we have already lemma 2. The proof of lemma 1 and its corollary are left as *exercices*: to prove lemma 1, do first an induction on the type α and then in the base case where $\alpha = N$ an induction on the definition of R_N. A predicate on expressions that satisfies the statement of lemma 1 is called *saturated*.

We prove now the key lemma 2. This is proved by induction on t:

- $t = x_i$, then we have $t\gamma \to u_i$; by hypothesis u_i is reducible, and hence $t\gamma$ is reducible *using lemma 1*,
- $t = 0$, then we have $0\gamma \to 0$, and hence $R_N\ (0\gamma)$ by definition of R_N,
- $t = S\ t'$, then we have $(S\ t')\gamma \to S\ (t'\gamma)$, but $t'\gamma$ is reducible by induction hypothesis, and hence so is $t\gamma$,
- $t = \lambda x\ t'$ is of type $\alpha \to \beta$, then we have $(\lambda x\ t')\gamma\ u \to t'(\gamma, x = u)$, for any reducible u of type α. By induction hypothesis $t'(\gamma, x = u)$ is reducible, and hence, *by lemma 1*, $t\gamma\ u$ is reducible,
- $t = t_1\ t_2$, then $t\gamma \to t_1\gamma\ (t_2\gamma)$; by induction hypothesis, both $t_1\gamma$ and $t_2\gamma$ are reducible, hence $t_1\gamma\ (t_2\gamma)$ is reducible and so is $t\gamma$ *using lemma 1*,
- $t = f_{a,b}$, then we prove that $t\gamma\ u$ is reducible by induction on the proof of $R_N\ u$. If $u \to^* 0$ then $t\gamma\ u \to^* a\gamma$. Since $a\gamma$ is reducible by induction hypothesis, then so is $t\gamma\ u$ *by* the corollary of lemma 1. If $u \to^* S\ u_1$ with u_1 reducible then $t\gamma \to^* b\gamma\ u_1\ (t\gamma\ u_1)$. By induction hypothesis, $t\gamma\ u_1$ is reducible; also by induction hypothesis (on t) $b\gamma$ is reducible. Since u_1 is reducible we can conclude *by* the corollary of lemma 1.

This proof is a good example of a proof by induction. There are two induction going on there: one is an induction on the types, and the other is the inductive definition of R_N.

The advantage of such a proof is that it is rather direct to extend it to the case of other data types. For instance, if we have a type of ordinals $O = 0 \mid S\ O \mid L\ (N \to O)$ the definition of being reducible of type O will be

$$\frac{u \to^* 0}{R_O\ u} \qquad \frac{u \to^* S\ u_1 \qquad R_O\ u_1}{R_O\ u} \qquad \frac{u \to^* L\ f \qquad (\forall v) R_N\ v \to R_O\ (f\ v)}{R_O\ u}.$$

1.3 Comment on $\vdash$ versus $\to$

It is interesting to compare the use of $\vdash$ and the use of $\to$ in this argument. One can consider $\to$ as an *internalisation* of $\vdash$, while the application of lambda-calculus is an internalisation of the operation of instantiation. The similarity in these two operations appear also in the framework of dependent types: if we write $[x : A]B$ for the dependent product and $\Gamma[x : A]$ the operation of extending a context with a type A then the rule of product formation can be formulated as: $\Gamma \vdash [x : A]B$ iff $\Gamma[x : A] \vdash B$.