Decidability in Intuitionistic Type Theory is functionally decidable

Silvio Valentini
Dipartimento di Matematica Pura ed Applicata
Università di Padova
via G. Belzoni n.7, I–35131 Padova, Italy
e-mail: valentini@pdmat1.math.unipad.it

September 24, 1996

Abstract

In this paper we show that the usual intuitionistic characterization of the decidability of the propositional function $B(x)$ *prop* $[x : A]$, i.e. to require that the predicate $(\forall x \in A)\ B(x) \vee \neg B(x)$ is provable, is equivalent, when working within the framework of Martin-Löf's Intuitionistic Type Theory, to require that there exists a *decision function* $\phi : A \to Boole$ such that $(\forall x \in A)\ (\phi(x) =_{Boole} true) \leftrightarrow B(x)$. Since we will also show that the proposition $x =_{Boole} true\ [x : Boole]$ is decidable, we can alternatively say that the main result of this paper is a proof that the decidability of the predicate $B(x)$ *prop* $[x : A]$ can be effectively reduced by a function $\phi \in A \to Boole$ to the decidability of the predicate $\phi(x) =_{Boole} true\ [x : A]$. All the proofs are carried out within the Intuitionistic Type Theory and hence the decision function ϕ, together with a proof of its correctness, is effectively constructed as a function of the proof of $(\forall x \in A)\ B(x) \vee \neg B(x)$.

1 The basic lemmas

The aim of this paper is to show that the usual intuitionistic characterization of the decidability of the propositional function $B(x)$ *prop* $[x : A]$, i.e. to require that the predicate $(\forall x \in A)\ B(x) \vee \neg B(x)$ is provable, is equivalent, when working within Martin-Löf's Intuitionistic Type Theory (ITT in the following), to require that there exists a decision function $\phi : A \to Boole$ such that $(\forall x \in A)\ (\phi(x) =_{Boole} true) \leftrightarrow B(x)$.

This result may not be completely new (for instance in a personal communication Martin-Löf said that he already knew it) but since, to my knowledge, there is no published material on this topic this note may be useful to a wider audience. In fact, apart from its intrinsical relevance, this result is also a good exercise in ITT since in order to be able to obtain its proof one has to use some of the most interesting properties of ITT. In this paragraph we will recall these

properties, and their proofs, to the reader who is not familiar with ITT, but to avoid to bore the reader who is familiar with ITT we will not recall all the basic definitions which can be found in [ML84] or [NPS90].

Since in most of the results the type *Boole* plays a central role, let us begin by recalling some of its properties. First of all recall that the canonical elements of the type *Boole* are *true* and $false$ and that, supposing $C(x)$ *prop* $[x : Boole]$, the *Boole*-elimination rule allows you to obtain $if\ c\ then\ d\ else\ e \in C(c)$ provided that $c \in Boole$, $d \in C(true)$ and $e \in C(false)$.

Lemma 1.1 *Let* $P(x)$ *prop* $[x : Boole]$ *and* $c \in Boole$*; then*

$$P(c) \to P(true) \vee P(false)$$

Proof. The proof is just an application of the *Boole*-elimination rule. In fact if $c \in Boole$ then $if\ c\ then\ \lambda x.inl(x)\ else\ \lambda y.inr(y) \in P(c) \to P(true) \vee P(false)$ since $\lambda x.inl(x) \in P(true) \to P(true) \vee P(false)$ and $\lambda y.inr(y) \in P(false) \to P(true) \vee P(false)$.

Let us recall that, supposing $c, d \in Boole$, by $c =_{Boole} d$ we mean the *Equality* proposition for elements of *Boole* whose main properties are that

- if $c = d \in Boole$ then $c =_{Boole} d$ is true,
- if $c =_{Boole} d$ is true and $A(x)$ *prop* $[x : Boole]$ is a proposition on elements of *Boole* such that $a \in A(c)$ then $move(c, a) \in A(d)$,
- if $c =_{Boole} d$ is true, $a(x) \in A$ $[x : Boole]$ and $y =_A z$ *prop* $[y, z : A]$ is the *Equality* proposition for elements of the type A then $a(c) =_A a(d)$ is true.

Hence the following corollary is immediate.

Corollary 1.2 *Let* $c \in Boole$*; then* $(c =_{Boole} true) \vee (c =_{Boole} false)$.

Proof. Suppose $P(x) \equiv c =_{Boole} x$ $[x : Boole]$, then the previous lemma shows that $(c =_{Boole} c) \to (c =_{Boole} true) \vee (c =_{Boole} false)$. Now the statement is obvious since $c =_{Boole} c$ is straightforward.

We have then proved that in *Boole* there are at most two elements; by means of the universe of the small types U_0, whose elements are (the codes of) the basic types, we can show that in *Boole* there are exactly two elements. To obtain this result it is convenient to use the *Equality* proposition $A =_{U_0} B$ *prop* $[A, B : U_0]$ for elements of the type U_0. Besides the properties analogous to those above for the *Equality* proposition for the elements of the type *Boole*, in this case one can also prove that if $A =_{U_0} B$ is true and $a \in A$ then $shift(a) \in B$.

Lemma 1.3 $\neg(true =_{Boole} false)$

Proof. Assume that $y : Boole$. Then a $Boole$-elimination can be used to shows that $if\ y\ then\ \top\ else\ \bot \in U_0$, where $\top$ is the one-element type and $\bot$ is the empty type. Let us now assume that $x : true =_{Boole} false$, i.e. let us assume that $true =_{Boole} false$ is true. Then we obtain that $if\ true\ then\ \top\ else\ \bot =_{U_0} if\ false\ then\ \top\ else\ \bot$ and hence $\top =_{U_0} \bot$ since the $Equality$ proposition is transitive and $\top =_{U_0} if\ true\ then\ \top\ else\ \bot$ and $if\ false\ then\ \top\ else\ \bot =_{U_0} \bot$ hold; hence, supposing $*$ is the only element of the type $\top$, $shift(*) \in \bot$, i.e. we have found an element in the empty type; so, by discharging the assumption $x : true =_{Boole} false$, we finally obtain $\lambda x.\ shift(*) \in \neg(true =_{Boole} false)$.

We showed a full detailed proof of this lemma to stress the fact that it is completely carried out within ITT with the universe U_0 of the small types.

Using lemma 1.3 and a little of intuitionistic logic one can prove the following not very surprising result.

Lemma 1.4 *$(c =_{Boole} false)$ if and only if $\neg(c =_{Boole} true)$.*

Even if the previous lemma is straightforward when we combine it with 1.2 we obtain an interesting result: the predicate $x =_{Boole} true\ [x : Boole]$ is decidable.

Corollary 1.5 *For all $c \in Boole$, $(c =_{Boole} true) \vee \neg(c =_{Boole} true)$.*

There is another property that we need to recall because of its relevance in the following: thanks to the constructive meaning of the logical connectives a sort of *Axiom of Choice* holds in ITT (here we show a statement which is not the strongest one that can be proved but it is sufficient for us).

Lemma 1.6 *Let A, B be two types and $C(x, y)$ prop $[x : A, y : B]$; then*

$$((\forall x \in A)(\exists y \in B)\ C(x, y)) \rightarrow ((\exists f \in A \rightarrow B)(\forall x \in A)\ C(x, f(x)))$$

Proof. A complete proof can be found in [ML84] where a choice function $f \in A \rightarrow B$ is constructed together with a formal proof that for any $x \in A$, $C(x, f(x))$ holds; anyhow the basic intuition to obtain the proof is rather simple: suppose h is (the code for) a proof of $(\forall x \in A)(\exists y \in B)\ C(x, y)$ then, for any $x \in A$, (the value of) $h(x)$ is a couple whose first element $p(h(x))$ belongs to B while the second element is a proof of $C(x, p(h(x)))$; the choice function is then $\lambda x.p(h(x)) \in A \rightarrow B$.

Also in this case we want to observe that the proof explicitly shows how to construct a choice function in $A \rightarrow B$ provided that we have a proof of $(\forall x \in A)(\exists y \in B)\ C(x, y)$.

2 The main result

This paragraph is completely devoted to the proof of the following theorem.

Theorem 2.1 *Let $B(x)$ prop $[x : A]$; then the following statements are equivalent:*

(1) There exists a decision function $\phi : A \to Boole$ such that, for all $x \in A$, $\phi(x) =_{Boole} true$ if and only if $B(x)$ is true.

(2) for all $x \in A$, $B(x) \vee \neg B(x)$.

We can straight away prove that (1) implies (2). In fact, let us suppose that $\phi : A \to Boole$ is a decision function for the proposition $B(x)$ *prop* $[x : A]$ and let us assume that $x \in A$. Then, because of corollary 1.5, we know that $(\phi(x) =_{Boole} true) \vee \neg(\phi(x) =_{Boole} true)$ and hence we can conclude $B(x) \vee \neg B(x)$ by $\vee$-elimination. In fact $\phi(x) =_{Boole} true$ immediately implies that $B(x)$ is true, and hence that $B(x) \vee \neg B(x)$ is true, since ϕ is a decision function. On the other hand the same conclusion can be obtained from the assumption $\neg(\phi(x) =_{Boole} true)$ by using the following derivation which again makes use of the fact that ϕ is a decision function:

$$\dfrac{\dfrac{\dfrac{[B(x)]_1}{\phi(x) =_{Boole} true} \quad \neg(\phi(x) =_{Boole} true)}{\dfrac{\bot}{\neg B(x)}\ 1}}{B(x) \vee \neg B(x)}$$

Let us now show that (2) implies (1); we do not only have to provide a function $\phi : A \to Boole$, which would be easy, but we have also to show that it is a decision function. This is the reason why we need some preliminary lemmas.

Lemma 2.2 *Let $B(x)$ prop $[x : A]$; then*

$$\begin{array}{l}(\forall x \in A)\ B(x) \vee \neg B(x) \to \\ \quad (\forall x \in A)(\exists y \in Boole) \\ \quad (y =_{Boole} true \to B(x))\ \&\ (y =_{Boole} false \to \neg B(x))\end{array}$$

Proof. The proof is just an application of $\vee$-elimination. In fact let us suppose that $(\forall x \in A)\ B(x) \vee \neg B(x)$ and assume that $x \in A$, then we have to show $(\exists y \in Boole)\ (y =_{Boole} true \to B(x))\ \&\ (y =_{Boole} false \to \neg B(x))$ from $B(x) \vee \neg B(x)$ and hence all we need are the following deductions:

$$\dfrac{\dfrac{B(x)}{true =_{Boole} true \to B(x)} \quad \dfrac{\neg(true =_{Boole} false)}{true =_{Boole} false \to \neg B(x)}}{\dfrac{(true =_{Boole} true \to B(x))\ \&\ (true =_{Boole} false \to \neg B(x))}{(\exists y \in Boole)\ (y =_{Boole} true \to B(x))\ \&\ (y =_{Boole} false \to \neg B(x))}}$$

and

$$\dfrac{\dfrac{\neg(false =_{Boole} true)}{false =_{Boole} true \to B(x)} \quad \dfrac{\neg B(x)}{false =_{Boole} false \to \neg B(x)}}{\dfrac{(false =_{Boole} true \to B(x))\ \&\ (false =_{Boole} false \to \neg B(x))}{(\exists y \in Boole)\ (y =_{Boole} true \to B(x))\ \&\ (y =_{Boole} false \to \neg B(x))}}$$

Thanks to lemma 2.2, we are in the position to take advantage of the axiom of choice in order to obtain the following corollary.

Corollary 2.3 *Let $B(x)$ prop $[x : A]$; then*

$$\begin{array}{l}(\forall x \in A)\ B(x) \vee \neg B(x) \rightarrow \\ \quad (\exists \phi \in A \rightarrow Boole)(\forall x \in A) \\ \quad (\phi(x) =_{Boole} true \rightarrow B(x)) \& (\phi(x) =_{Boole} false \rightarrow \neg B(x))\end{array}$$

Hence we have obtained the proof of the main theorem; in fact this corollary shows that if we have a proof of $(\forall x \in A)\ B(x) \vee \neg B(x)$ then both

(†) there exists a function $\phi : A \rightarrow Boole$, and since all the proofs were developed within ITT we can effectively construct it, and

(‡) such a function is a decision function for $B(x)$; in fact if $\phi(x) =_{Boole} true$ then $B(x)$ is true and, on the other hand, if $B(x)$ is true we can use the following derivation to show that $\phi(x) =_{Boole} true$:

$$\dfrac{\phi(x) =_{Boole} true \vee \phi(x) =_{Boole} false \qquad \dfrac{\dfrac{[\phi(x) =_{Boole} false]_1 \quad \phi(x) =_{Boole} false \rightarrow \neg B(x)}{\neg B(x)} \qquad B(x)}{\dfrac{\bot}{\neg(\phi(x) =_{Boole} false)}\ 1}}{\phi(x) =_{Boole} true}$$

We can save a curious reader the trouble of doing some work if we say that, supposing $h \in (\forall x \in A)\ B(x) \vee \neg B(x)$, the decision function which we obtain (after some unessential simplification) is

$$\begin{array}{lll}\phi \equiv & \lambda x.\ p(D(& h(x), \\ & & (z)\ < true, < \lambda w.\ z, \lambda u.\ R_0(shift(*)) >>, \\ & & (z)\ < false, < \lambda u.\ R_0(shift(*)), \lambda w.\ z >>)).\end{array}$$

We can simplify it by far if we disregard all the parts which do not have a computational content and which appear in ϕ only because of the way we obtained it; in fact the function

$$\phi' \equiv \lambda x. D(h(x), (z)\ true, (z)\ false)$$

has obviously the same computational behavior as ϕ; the drawback is that we lack a *formal* proof that ϕ' is a decision function for $B(x)$. Of course we can obtain such a proof by using the fact that ϕ *is* a decision function for $B(x)$. In fact we can prove that $(\forall x \in A)\ \phi(x) =_{Boole} \phi'(x)$ is true, because in general, supposing A, B, C, D are types, $c \in A + B$, $d(z) \in C \times D\ [z : A]$, $e(z) \in C \times D\ [z : B]$, the *Equality* proposition $p(D(c, (z)\ d(z), (z)\ e(z)) =_C D(c, (z)\ p(d(z)), (z)\ p(e(z)))$ holds.

References

[ML84] P. Martin-Löf, *Intuitionistic Type Theory, Notes by G. Sambin of a series of lectures given in Padua*, Bibliopolis, 1984

[NPS90] B. Nordström, K. Peterson, J. Smith, *Programming in Martin-Löf's Type Theory, An introduction*, Clarendon Press, Oxford, 1990