The Strength of the Subset Type in Martin-Löf's Type Theory

Anne Salvesen *Jan M. Smith*

March 1988

1 Introduction

A program satisfying a specification in Martin-Löf's type theory may have parts which never will be used when computing the program. The reason for this is that the interpretation of propositions as types may force programs to contain proof objects which only serve as witnesses of the truth of some propositions which are required to hold by the specification. The subset types have been introduced in type theory in order to avoid this problem [6]. Because of the interpretation of propositions as types, which demands that we must have an explicit proof object when we express the truth of a proposition, it is not obvious how to formulate rules for subsets in type theory. The formation and introduction rules are what one expects:

Subset-formation

$$\frac{A\ \mathit{type} \qquad B(x)\ \mathit{type}\ [x \in A]}{\{x \in A \mid B(x)\}\ \mathit{type}}$$

Subset-introduction

$$\frac{a \in A \qquad b \in B(a)}{a \in \{x \in A \mid B(x)\}}$$

An elimination rule which captures the way we have introduced objects of a subset is impossible to give in type theory because when we have an object a of a subset $\{x \in A \mid B(x)\}$ we have no explicit construction of the proof object of $B(a)$. The best formulation of an elimination rule we can give is the following:

Subset-elimination

$$\frac{a \in \{x \in A \mid B(x)\} \qquad c(x) \in C(x)\ [x \in A,\ y \in B(x)]}{c(a) \in C(a)}$$

where y must not occur free in $c(x)$ nor in $C(x)$

In this rule it is required that $C(x)$ is a type under the assumption $x \in \{z \in A \mid B(z)\}$. We will in this paper show that the exact formulation of the rules of type theory is very important for the power of the subset type; it actually turns out that there are propositions involving subsets which are trivially true in naive set theory but which cannot be proved in type theory. We will look at the provability of propositions of the form

$$(\forall x \in \{z \in A \mid P(z)\})P(x) \qquad (*)$$

Propositions of this form are important when modularizing program derivations, using a top-down approach and decomposing the specification into subproblems. When solving the subproblems we may want to use lemmas which have already been proved. The main idea of splitting up a problem into lemmas is, in program derivation as well as in mathematics, that our original problem can be reduced to the lemmas; in particular, there should be no need to look into the proofs of the lemmas. If we have a lemma which talks about subsets we certainly want $(*)$ to be provable since if $a \in \{x \in A \mid P(x)\}$ we want to be able to conclude $P(a)$ without having to investigate the proof of $a \in \{x \in A \mid P(x)\}$.

We will discuss adding subsets to two formulations of Martin-Löf's type theory which differ in the equality type. In the first formulation, ITT, the judgemental equality $a = b \in A$ is understood as intensional, or definitional, equality. The rules for the equality types are those in Martin-Löf [3] which also are the rules now used by Martin-Löf. The other formulation, ETT, is that of Martin-Löf [4, 5], where the judgemental equality is extensional and undecidable. The type theory used in Nuprl [1] is based on ETT.

Because of the syntactical restriction on free variables in the subset-elimination rule, the strength of this rule is connected with the possibility of having rules in type theory where free variables, other than those discharged by the rule, may disappear in the conclusion. In ITT there are actually no essential possibilities to get rid of free variables. Hence, when adding subsets to ITT, the subset-elimination rule is limited which, as we will see, results in a very weak subset type for which $(*)$ can only be proved in some trivial cases. In ETT there is a possibility to get rid of variables in the rules for the equality type. So in ETT there are more cases when we can use subset-elimination and we will give a general condition on the predicate $P(x)$ which will imply the provability of $(*)$. Nevertheless, in section 5 we will give an instance of $(*)$ which cannot be proved in type theory irrespectively of how we formulate the rules; the example only requires that the axiom of choice, as formulated in [4, 5], can be proved and that a typable term can be computed by a Turing machine.

2 The subset type in ITT

In ITT the judgemental equality is understood as definitional equality and the equality type reflects this. So the objects of the equality type are introduced by:

Eq-introduction

$$\frac{a \in A}{\mathsf{eq}(a) \in \mathsf{Eq}(A, a, a)}$$

and the elimination rule gives a corresponding principle of structural induction over an equality type:

Eq-elimination

$$\frac{c \in \mathsf{Eq}(A, a, b) \qquad d(x) \in C(x, x, \mathsf{eq}(x))\ [x \in A]}{\mathsf{eqpeel}(c, d) \in C(a, b, c)}$$

The rule of reduction for the noncanonical form introduced in this rule is that the expression $\mathsf{eqpeel}(\mathsf{eq}(a), d)$ reduces to $d(a)$. So, in general, the argument a in $\mathsf{eq}(a)$ is needed when computing $\mathsf{eqpeel}(\mathsf{eq}(a), d)$.

Note that the canonical expression $\mathsf{eq}(a)$ in the conclusion of the Eq-introduction rule contains the object expression a in the premiss and that the type expression $\mathsf{Eq}(A, a, a)$ in the conclusion contains the type expression A in the premiss. In particular, all free variables occurring in the premiss also occur free in the conclusion. This also holds for all the other introduction rules in ITT; that is, all free variables occurring in the object and the type of a premiss, except those discharged by the rule, also occur free in the object and type of the conclusion, respectively. This fact is used in a crucial way in the proof of the following theorem, which will enable us to give very simple instances of $(*)$ which cannot be proved in ITT.

Before formulating the theorem, we define a mapping $'$ which to each object expression a in ITT with subsets associates an object expression a' in ITT and to each type expression A in ITT with subsets associates a type expression A' in ITT. The mapping is defined so that it commutes with all constants in ITT. For subsets it is defined by

$$\{x \in A \mid B(x)\}' = A'$$

and for codes for subsets in the universe U it is correspondingly defined by

$$\widehat{\{|\}}\,(a, b)' = a'$$

Theorem 1 *If $a \in A$ is derivable in ITT extended with the rules for subsets, then a' can be computed to a canonical value of type A'.*

Since, by the definition of the mapping $'$,

$$((\forall x \in \{z \in A \mid P(z)\})P(x))' = (\forall x \in A')P(x)'$$

we obtain the following corollary from theorem 1:

Corollary 1 *If $t \in (\forall x \in \{z \in A \mid P(z)\})P(x)$ is derivable in* ITT *with the rules for subsets, and neither A nor P syntactically contains the subset type, then t' can be computed to a canonical value of type $(\forall x \in A)P(x)$.*

For instance, not even $(\forall x \in \{z \in \mathsf{T} \mid \perp\})\perp$ can be proved in ITT with subsets, since otherwise we would have, by the corollary, a term t' which could be computed to a canonical value of type $(\forall x \in \mathsf{T})\perp$ Hence, $\mathsf{apply}(t', \mathsf{tt})$, where tt is the canonical object in T, could be computed to a canonical value of type $\perp$ which is impossible since $\perp$ is empty. So the corollary is a strongly negative result which shows that a straightforward introduction of the subset type by just adding formation, introduction and elimination rules does not work for ITT.

Proof of theorem 1. The proof of this theorem is by induction on the length of the derivation of $a \in A$ and is based on Tait's method [10] for proving normalization. However, in order to cope with subset-elimination, where the interpretation by $'$ of the type in the major premiss $a \in \{x \in A \mid B(x)\}$ is just A', we must show that we in the interpretation of the minor premiss $c(x) \in C(x)$ $[x \in A,\ y \in B(x)]$ can do without the assumption that there exists a canonical element in $B(a)'$ when concluding that $c(a)'$ can be computed to a canonical value of type $C(a)'$; something which does not hold for ETT. We must therefore, compared with the normalization proof in Martin-Löf [3], strengthen the induction hypothesis by having stronger conditions on the computable terms of a type depending on assumptions. We will not give all the details of the proof, but concentrate on the differences compared with [3].

The set $\mathsf{Comp}(A)$ of computable terms of type A in a context Γ_A is defined by induction on the length of the derivation of A *type* $[\Gamma_A]$ in ITT; in particular, the induction hypothesis implies that the set of computable terms has been defined for all the types in the context Γ_A. We define the set of computable terms for N, the function types and equality types; the remaining types are handled in a similar way.

We are using the notation $e(e_1/x_1, \ldots, e_n/x_n)$ for the result of substituting $e_1, \ldots, e_n$ for the free variables $x_1, \ldots, x_n$ in e, respectively, where e may either be an ordinary expression or a context.

That the term t is an element in $\mathsf{Comp}(\mathsf{N})$ in the context Γ_N is defined in the following way:

The free variables of t must all be declared in the context Γ_N. Let

$$\Gamma_t \;\equiv\; y_1 \in C_1,\ \ldots,\ y_n \in C_n(y_1, \ldots, y_{n-1})$$

be the smallest subcontext of Γ_N such that all the free variables in t are declared in Γ_t. The term t is an element in the set

$$\mathsf{Comp}(\mathsf{N})$$

if for all $c_1 \in \mathsf{Comp}(C_1),\ \ldots,\ c_n \in \mathsf{Comp}(C_n(c_1,\ldots,c_{n-1}))$ the term $t(c_1/y_1,\ldots,c_n/y_n)$ can be computed to a numeral, that is, it has either value $\mathsf{0}$ or a value of the form $\mathsf{succ}(\mathsf{succ}(\ldots(\mathsf{0})\ldots))$.

We must also define what it means for two elements s and t in $\mathsf{Comp}(\mathsf{N})$ to equal in $\mathsf{Comp}(\mathsf{N})$ in the context Γ_N. Let

$$\Gamma_{st} \ \equiv\ y_1 \in C_1,\ \ldots,\ y_n \in C_n(y_1,\ldots,y_{n-1})$$

be the smallest subcontext of Γ_N such that all the free variables in s and t are declared in Γ_t. The terms s and t are equal in the set

$$\mathsf{Comp}(\mathsf{N})$$

if for all $c_1 \in \mathsf{Comp}(C_1),\ \ldots,\ c_n \in \mathsf{Comp}(C_n(c_1,\ldots,c_{n-1}))$ the terms $s(c_1/y_1,\ldots,c_n/y_n)$ and $t(c_1/y_1,\ldots,c_n/y_n)$ have the same numeral as value.

We now come to the definition of the set of computable terms for a function type. Let A be a type in a context Γ_A and B a type in a context Γ_B. Then $A \to B$ is a type in a context $\Gamma_{A\to B}$ containing Γ_A and Γ_B and, by the induction hypothesis, the set of computable terms has been defined for A and B as well as for all the types appearing in the context $\Gamma_{A\to B}$. That a term t is an element in $\mathsf{Comp}(A \to B)$, depending on the context $\Gamma_{A\to B}$, is defined in the following way:

The free variables of the term t must all be declared in the context $\Gamma_{A\to B}$. Let

$$\Gamma_t \ \equiv\ y_1 \in C_1,\ \ldots,\ y_n \in C_n(y_1,\ldots,y_{n-1})$$

be the smallest subcontext of $\Gamma_{A\to B}$ such that all the free variables in t are declared in Γ_t. The term t is an element in the set

$$\mathsf{Comp}(A \to B)$$

if for all $c_1 \in \mathsf{Comp}(C_1),\ \ldots,\ c_n \in \mathsf{Comp}(C_n(c_1,\ldots,c_{n-1}))$ there exists a term $b(x)$ such that $t(c_1/y_1,\ldots,c_n/y_n)$ can be computed to $\lambda x.b(x)$ and where $b(x)$ has the property given below.

Let a be a term with its free variables declared in Γ_A and which has the following property:

Let

$$\Gamma_a \ \equiv\ u_1 \in D_1,\ \ldots,\ u_k \in D_k(u_1,\ldots,u_{k-1})$$

be the smallest subcontext of $\Gamma_A(c_1/y_1,\ldots,c_n/y_n)$ such that all free variables in $a(c_1/y_1,\ldots,c_n/y_n)$ are declared in Γ_a.

Let $d_1 \in \mathsf{Comp}(D_1),\ \ldots,\ d_k \in \mathsf{Comp}(D_k(d_1,\ldots,d_{k-1}))$ and let

$$v_1 \in E_1,\ \ldots,\ v_l \in E_l(v_1,\ldots,v_{l-1})$$

be the smallest context containing the free variables in

$$A(c_1/y_1,\ldots,c_n/y_n,\, d_1/u_1,\ldots,d_k/u_k)$$

Then

$$a(c_1/y_1,\ldots,c_n/y_n,\, d_1/u_1,\ldots,d_k/u_k) \in \\ \mathsf{Comp}(A(c_1/y_1,\ldots,c_n/y_n,\, d_1/u_1,\ldots,d_k/u_k,\, e_1/v_1,\ldots,e_l/v_l))$$

for all $e_1 \in \mathsf{Comp}(E_1),\ \ldots,\ e_l \in \mathsf{Comp}(E_l(e_1,\ldots,e_{n-1}))$

Now we come to the property that $b(x)$ must satisfy. Let

$$z_1 \in F_1,\ \ldots,\ z_p \in F_p(z_1,\ldots,z_{p-1})$$

be the smallest context containing the free variables in

$$B(c_1/y_1,\ldots,c_n/y_n,\, d_1/u_1,\ldots,d_k/u_k)$$

Then

$$b(a(c_1/y_1,\ldots,c_n/y_n,\, d_1/u_1,\ldots,d_k/u_k)) \in \\ \mathsf{Comp}(B(c_1/y_1,\ldots,c_n/y_n,\, d_1/u_1,\ldots,d_k/u_k,\, f_1/z_1,\ldots,f_p/z_p))$$

for all $f_1 \in \mathsf{Comp}(F_1),\ \ldots,\ f_p \in \mathsf{Comp}(F_p(f_1,\ldots,f_{p-1}))$.

The difference between this definition of $\mathsf{Comp}(A \to B)$ and the definition in Martin-Löf [3] is that in [3] it is required that the term t must have a value of the form $\lambda x.b(x)$ only when we substitute computable terms for all the variables in the context $\Gamma_{A\to B}$.

We must also define equality on $\mathsf{Comp}(A \to B)$. So let s and t be elements in $\mathsf{Comp}(A \to B)$ and let

$$\Gamma_{st} \ \equiv\ y_1 \in C_1,\ \ldots,\ y_n \in C_n(y_1,\ldots,y_{n-1})$$

be the smallest subcontext of $\Gamma_{A\to B}$ such that all free variables in s and t are declared in Γ_{st}.

The terms s and t are equal in $\mathsf{Comp}(A \to B)$ if for all $c_1 \in \mathsf{Comp}(C_1),\ \ldots,\ c_n \in \mathsf{Comp}(C_n(c_1,\ldots,c_{n-1}))$ the value $\lambda x.b_s(x)$ of $s(c_1/y_1,\ldots,c_n/y_n)$ and the value $\lambda x.b_t(x)$ of $t(c_1/y_1,\ldots,c_n/y_n)$ have the property given below.

Let a be a term which satisfy similar conditions as the term a in the definition of the elements in $\mathsf{Comp}(A \to B)$. So a is a term with its free variables declared in Γ_A and which has the following property:

Let

$$\Gamma_a \ \equiv\ u_1 \in D_1,\ \ldots,\ u_k \in D_k(u_1,\ldots,u_{k-1})$$

be the smallest subcontext of $\Gamma_A(c_1/y_1,\ldots,c_n/y_n)$ such that all free variables in $a(c_1/y_1,\ldots,c_n/y_n)$ are declared in Γ_a.

Let $d_1 \in \mathsf{Comp}(D_1),\ \ldots,\ d_k \in \mathsf{Comp}(D_k(d_1,\ldots,d_{k-1}))$ and let

$$v_1 \in E_1,\ \ldots,\ v_l \in E_l(v_1,\ldots,v_{l-1})$$

be the smallest context containing the free variables in

$$A(c_1/y_1,\ldots,c_n/y_n,\, d_1/u_1,\ldots,d_k/u_k)$$

Then

$$a(c_1/y_1, \ldots, c_n/y_n,\, d_1/u_1, \ldots, d_k/u_k) \in \\ \mathsf{Comp}(A(c_1/y_1, \ldots, c_n/y_n,\, d_1/u_1, \ldots, d_k/u_k,\, e_1/v_1, \ldots, e_l/v_l))$$

for all $e_1 \in \mathsf{Comp}(E_1),\ \ldots,\ e_l \in \mathsf{Comp}(E_l(e_1, \ldots, e_{n-1}))$

Now we come to the property that $b_s(x)$ and $b_t(x)$ must satisfy. Let

$$z_1 \in F_1,\ \ldots,\ z_p \in F_p(z_1, \ldots, z_{p-1})$$

be the smallest context containing the free variables in

$$B(c_1/y_1, \ldots, c_n/y_n,\, d_1/u_1, \ldots, d_k/u_k)$$

Then

$$b_s(a(c_1/y_1, \ldots, c_n/y_n,\, d_1/u_1, \ldots, d_k/u_k))$$

and

$$b_t(a(c_1/y_1, \ldots, c_n/y_n,\, d_1/u_1, \ldots, d_k/u_k))$$

are equal elements in

$$\mathsf{Comp}(B(c_1/y_1, \ldots, c_n/y_n,\, d_1/u_1, \ldots, d_k/u_k,\, f_1/z_1, \ldots, f_p/z_p))$$

for all $f_1 \in \mathsf{Comp}(F_1),\ \ldots,\ f_p \in \mathsf{Comp}(F_p(f_1, \ldots, f_{p-1}))$.

We will also define $\mathsf{Comp}(\mathsf{Eq}(A,a,b))$. So let A be a type in a context Γ_A, a an object of A in a context Γ_a and b an object of B in a context Γ_b. Then $\mathsf{Eq}(A,a,b)$ is a type in a context $\Gamma_{Eq(A,a,b)}$ containing the contexts Γ_A, Γ_a and Γ_b. That the term t is an element in $\mathsf{Comp}(\mathsf{Eq}(A,a,b))$, depending on the context $\Gamma_{Eq(A,a,b)}$ is defined in the following way:

Let the free variables of the term t all be declared in the context $\Gamma_{Eq(A,a,b)}$. Let

$$\Gamma_t \ \equiv\ y_1 \in C_1,\ \ldots,\ y_n \in C_n(y_1, \ldots, y_{n-1})$$

be the smallest subcontext of $\Gamma_{Eq(A,a,b)}$ containing all the free variables in t. The term t is an element in

$$\mathsf{Comp}(\mathsf{Eq}(A,a,b))$$

if for all $c_1 \in \mathsf{Comp}(C_1),\ \ldots,\ c_n \in \mathsf{Comp}(C_n(c_1, \ldots, c_{n-1}))$ there exists a term e such that $t(c_1/y_1, \ldots, c_n/y_n)$ can be computed to $\mathsf{eq}(e)$ where e has the following property:

Let

$$u_1 \in A_1,\ \ldots,\ u_k \in A_k(u_1, \ldots, u_{k-1})$$

be the context $\Gamma_A(c_1/y_1, \ldots, c_n/y_n)$. Then

$$e \in \mathsf{Comp}(A(c_1/y_1, \ldots, c_n/y_n,\, a_1/u_1, \ldots, a_k/u_k))$$

for all $a_1 \in \mathsf{Comp}(A_1),\ \ldots,\ a_k \in \mathsf{Comp}(A_k(a_1, \ldots, a_{k-1}))$ and $a(c_1/y_1, \ldots, c_n/y_n)$ and $b(c_1/y_1, \ldots, c_n/y_n)$ can both be computed to e.

Following the pattern in $\mathsf{Comp}(A \to B)$, it is straightforward to define what it means for two elements to be equal in $\mathsf{Comp}(\mathsf{Eq}(A, a, b))$.

Let the context Γ' be obtained from Γ by applying the mapping $'$ on all the types in Γ. We can now prove, by induction on the length of the derivation, that

- if A *type* is derivable in a context Γ in ITT with subsets, then $\mathsf{Comp}(A')$ is defined in the smallest subcontext of Γ' containing all the free variables in A'.
- if $A = B$ is derivable in a context Γ in ITT with subsets, then $\mathsf{Comp}(A') = \mathsf{Comp}(B')$ in the smallest subcontext of Γ' containing all the free variables in A' and B'.
- if $a \in A$ is derivable in a context Γ in ITT with subsets, then $a' \in \mathsf{Comp}(A')$ in the smallest subcontext of Γ' containing all the free variables in a' and A'.
- if $a = b \in A$ is derivable in a context Γ in ITT with subsets, then a' and b' are equal elements in $\mathsf{Comp}(A')$ in the smallest subcontext of Γ' containing all the free variables in a, b, and A.

The proof is trivial in the case of an introduction rule because of the remark above that no free variables, except those possibly discharged by the rule, disappears in the conclusion of an introduction rule in ITT. We do the induction step for two of the elimination rules.

Subset-elimination

$$\frac{a \in \{x \in A \mid B(x)\} \qquad c(x) \in C(x)\ [x \in A,\ y \in B(x)]}{c(a) \in C(a)}$$

By the induction hypothesis, we know that $a' \in \mathsf{Comp}(\{x \in A \mid B(x)\}')$. Since y does not occur free in $c(x)$, we know that $c'(d) \in \mathsf{Comp}(C'(d))$ for all $d \in \mathsf{Comp}(A')$ because $C(x)$ is a type under the assumption $x \in \{z \in A \mid B(z)\}$ and $\{x \in A \mid B(x)\}' \equiv A'$. Hence, $c'(a') \in \mathsf{Comp}(C'(a'))$ as desired.

We also verify a crucial instance of $\to$-elimination in which the conclusion $\mathsf{apply}(f, a) \in B$ depends on an assumption $y \in C$ although y occurs free neither in $\mathsf{apply}(f, a)$ nor in B.

$$\frac{a \in A(y)\ [y \in C] \qquad f \in A(y) \to B\ [y \in C]}{\mathsf{apply}(f, a) \in B\ [y \in C]}$$

We are here assuming that $A(y)$ *type* $[y \in C]$, B *type* and that y is occurring free neither in a nor in f. By the induction hypothesis, $f' \in \mathsf{Comp}(A'(y) \to B')$ in the context $y \in C$ which means that f' can be computed to a term $\lambda x.b(x)$ such that if $a \in \mathsf{Comp}(A'(c))$ for all $c \in \mathsf{Comp}(C')$ then $b(a) \in \mathsf{Comp}(B')$. By the induction hypothesis, we have $a' \in \mathsf{Comp}(A'(c))$ for all $c \in \mathsf{Comp}(C')$. Hence, since $\mathsf{apply}(f', a')$ reduces to $b(a')$, we get $\mathsf{apply}(f', a') \in \mathsf{Comp}(B')$.

□

The main part of this proof is really a normalization proof for ITT, which, because of the strong definition of the computability predicate, is of interest already without subsets. A more detailed presentation of this proof together with some consequences of it for ITT will be given in a paper in preparation by Jan Smith.

Remark. The exact formulation of ITT we have in mind here is obtained from the formulation of type theory in [4, 5] by just replacing the rules for the equality type. This formulation of type theory is polymorphic, but if we instead consider a monomorphic formulation of ITT, we can strengthen theorem 1 to even get that if we have a closed derivation of $a \in A$ then $a' \in A'$ is also derivable. The proof is by induction on the length of the derivation of $a \in A$ and we then also have to consider open derivations. The crucial lemma is then that if $a \in A(y)\ [y \in C]$ is derivable in the monomorphic ITT and a does not contain y free, then there exists a type A^* not contain y free and such that $A^* = A(y)\ [y \in C]$ and $a \in A^*$. The proof of this lemma, which does not hold for the polymorphic formulation of ITT, uses the Church-Rosser property. The reason why the theorem can be strengthened for a monomorphic theory is that we now cannot even in an elimination rule lose any variables.

3 The subset type in ETT

In ETT the canonical objects of the equality type are introduced by reflection of the judgemental equality:

Eq-introduction

$$\frac{a = b \in A}{\mathsf{eq} \in \mathsf{Eq}(A, a, b)}$$

There is an elimination rule giving a structural induction principle, but the important rule is the strong Eq-elimination which connects the judgemental level with the type level:

Strong Eq-elimination

$$\frac{c \in \mathsf{Eq}(A, a, b)}{a = b \in A}$$

This elimination rule is much stronger than the Eq-elimination rule in ITT since judgemental equalities now can be proved by mathematical reasoning, using the propositional level of type theory.

Let

$$P(x_1, \ldots, x_n)\ \mathit{type}\ [x_1 \in A_1,\ \ldots,\ x_n \in A_n(x_1, \ldots, x_{n-1})]$$

The predicate $P(x_1, \ldots, x_n)$ is called *stable* if

$$\neg\neg P(x_1, \ldots, x_n) \rightarrow P(x_1, \ldots, x_n) \\ [x_1 \in A_1, \ldots,\ x_n \in A_n(x_1, \ldots, x_{n-1})]$$

Using strong Eq-elimination together with the universe, we can prove that $(*)$ holds for all stable predicates:

Theorem 2 *In* ETT *extended with the rules for subsets, we can derive* $(\forall x \in A)(\neg\neg P(x) \rightarrow P(x)) \rightarrow (\forall x \in \{z \in A \mid P(z)\})P(x)$.

Proof. By the interpretation of propositions as types, we have to construct an object of the type

$$(\Pi x \in \{z \in A \mid P(z)\})P(x) \tag{1}$$

from the assumption

$$h \in (\Pi x \in A)(((P(x) \rightarrow \perp) \rightarrow \perp) \rightarrow P(x)) \tag{2}$$

We will derive (1) by constructing an object

$$p(x) \in P(x) \quad [x \in \{z \in A \mid P(z)\}] \tag{3}$$

and then use Π-introduction. We will first derive

$$p(z) \in P(z) \quad [z \in A,\ y \in P(z)]$$

from which we then obtain (3) by subset-elimination. Of course, $p(z)$ must not contain the variable y free. In order to construct $p(z)$ we will construct an object

$$q(z) \in (P(z) \rightarrow \perp) \rightarrow \perp \quad [z \in A,\ y \in P(z)]$$

such that $q(z)$ does not contain the variable y free and then use the assumption (2).

The first step in the derivation is to apply $\rightarrow$-elimination on the assumptions $y \in P(z)$ and $u \in P(z) \rightarrow \perp$ to obtain

$$\mathsf{apply}(u, y) \in \perp \quad [z \in A,\ y \in P(z),\ u \in P(z) \rightarrow \perp] \tag{4}$$

The term $\mathsf{apply}(u, y)$ contains y free, so we will now construct an object in $\perp$ which does not contain y free. By $\perp$-elimination applied on (4), we obtain

$$\mathsf{case}_0(\mathsf{apply}(u, y)) \in \mathsf{Eq}(\mathsf{U}, \widehat{\mathsf{N}}, \widehat{\perp}) \quad [z \in A,\ y \in P(z),\ u \in P(z) \rightarrow \perp] \tag{5}$$

where U is the first universe, i.e. the type of encodings of the small types, and $\widehat{\mathsf{N}}$ and $\widehat{\perp}$ are the codes for N and $\perp$, respectively. The strong Eq-elimination rule applied on (5) gives

$$\widehat{\mathsf{N}} = \widehat{\perp} \in \mathsf{U} \quad [z \in A,\ y \in P(z),\ u \in P(z) \rightarrow \perp]$$

from which we get, by the decoding rules for the universe,

$$\mathsf{N} = \bot \quad [z \in A,\ y \in P(z),\ u \in P(z) \to \bot] \tag{6}$$

Since $\mathsf{0} \in \mathsf{N}$ we get from (6)

$$\mathsf{0} \in \bot \quad [z \in A,\ y \in P(z),\ u \in P(z) \to \bot] \tag{7}$$

and we have obtained an object in $\bot$ not containing the variable y free. $\to$-introduction on (7) gives

$$\lambda u.\mathsf{0} \in (P(z) \to \bot) \to \bot \quad [z \in A,\ y \in P(z)] \tag{8}$$

From the assumption (2) we get, by Π-elimination,

$$\mathsf{apply}(h, z) \in ((P(z) \to \bot) \to \bot) \to P(z) \quad [z \in A] \tag{9}$$

Using $\to$-elimination on (8) and (9) we obtain

$$\mathsf{apply}(\mathsf{apply}(h, z), \lambda u.\mathsf{0}) \in P(z) \quad [z \in A,\ y \in P(z)] \tag{10}$$

Since y does not occur free in $\mathsf{apply}(\mathsf{apply}(h, z), \lambda u.\mathsf{0})$ we can apply subset-elimination on (10) and obtain

$$\mathsf{apply}(\mathsf{apply}(h, x), \lambda u.\mathsf{0}) \in P(x) \quad [x \in \{z \in A \mid P(z)\}]$$

and, by using Π-introduction, we finally obtain

$$\lambda x.\mathsf{apply}(\mathsf{apply}(h, x), \lambda u.\mathsf{0}) \in (\Pi x \in \{z \in A \mid P(z)\})P(x)$$

$\square$

The interest of theorem 2 depends on how large the class of stable propositions is. Using the strong Eq-elimination rule in a crucial way, we can prove that $\mathsf{Eq}(A, x, y)$ is stable:

Theorem 3 $(\forall x \in A)(\forall y \in A)(\neg\neg\mathsf{Eq}(A, x, y) \to \mathsf{Eq}(A, x, y))$ *can be derived in* ETT *for all types* A.

Proof. The proof is by induction of the length of the derivation that A is a type. It turns out that the proof can be carried out in a uniform manner for all the formation rules as well as for the rule

$$\frac{a \in \mathsf{U}}{\mathsf{Type}(a) \ \ \textit{type}}$$

where we also must use U-induction, reflecting the proof for the formation rules.

The key to the proof lies in the rules for introducing equal canonical objects of the types together with the strong Eq-elimination rule. Let A be a type obtained by one of the formation rules. It is easy to refine our goal to a subgoal where the objects are on canonical form. Hence, our original goal is achieved if we can find an object of

$$\mathsf{Eq}(A,a,b)\ [I_A, p \in \neg\neg\mathsf{Eq}(A,a,b)] \tag{1}$$

where I_A is a list of assumptions, depending on the type A, and a and b are on canonical form. There are two cases: a and b have different outer canonical form or a and b have equal outer canonical form. The first case is easily proved by absurdity-elimination. In the second case, $p \in \neg\neg\mathsf{Eq}(A,a,b)$ and the induction hypothesis will allow us to construct a proof of the premisses of the introduction rule for equal canonical objects of the type A. It is then straightforward to obtain a proof of (1). We will give the details of the proof in the case when A has been introduced by Π-formation:

$$\frac{C\ \mathit{type} \qquad D(z)\ \mathit{type}\ [z \in C]}{(\Pi z \in C)D(z)\ \mathit{type}}$$

By the induction hypothesis we can find objects f_1 and f_2 such that

$$f_1 \in (\Pi x \in C)(\Pi y \in C)(\neg\neg\mathsf{Eq}(C,x,y) \to \mathsf{Eq}(C,x,y))$$

and

$$\begin{aligned} f_2 \in (\Pi x \in D(z))&(\Pi y \in D(z)) \\ &(\neg\neg\mathsf{Eq}(D(z),x,y) \to \mathsf{Eq}(D(z),x,y))\ [z \in C] \end{aligned} \tag{2}$$

We must construct an object of the type

$$\begin{aligned} (\forall x \in (\Pi z \in C)&D(z))(\forall y \in (\Pi z \in C)D(z)) \\ &(\neg\neg\mathsf{Eq}((\Pi z \in C)D(z),x,y) \to \mathsf{Eq}((\Pi z \in C)D(z),x,y)) \end{aligned}$$

Refining this goal with introduction and elimination with respect to both quantifiers leaves us with the goal

$$\begin{aligned} \neg\neg\mathsf{Eq}((\Pi z \in C)D(z), \lambda z.b(z), \lambda z.d(z)) &\to \\ \mathsf{Eq}((\Pi z \in C)D(z), \lambda z.b(z), \lambda z.d(z))& \end{aligned} \tag{3}$$

under the list of assumptions I_Π:

$$\begin{aligned} &h_1 \in (\Pi z \in C)D(z),\ b(x) \in D(x)\ [x \in C], \\ &\qquad h_2 \in (\Pi z \in C)D(z),\ d(x) \in D(x)\ [x \in C] \end{aligned}$$

Note that we have here used the strong formulation of Π-elimination discussed in the preface of [5]. We will prove (3) by constructing an object of the type

$$\neg\neg\mathsf{Eq}(D(z), b(z), d(z)) \tag{4}$$

under the assumptions

$$I_\Pi,\ g \in \neg\neg\mathsf{Eq}((\Pi z \in C)D(z), \lambda z.b(z), \lambda z.d(z)),\ z \in C$$

We can then use f_2 on (4) to get an object of

$$\mathsf{Eq}(D(z), b(z), d(z))$$

and then strong Eq-elimination to obtain

$$b(z) = d(z) \in D(z) \tag{5}$$

under the assumptions

$$I_\Pi,\ g \in \neg\neg\mathsf{Eq}((\Pi z \in C)D(z), \lambda z.b(z), \lambda z.d(z)),\ z \in C$$

Using the rule

$$\frac{b(x) = d(x) \in D(x)\,[x \in C]}{\lambda x.b(x) = \lambda x.d(x) \in (\Pi x \in C)D(x)}$$

for forming equal canonical objects of the Π-type, we obtain (3) from (5) by Eq-introduction and $\rightarrow$-introduction. In order to prove (4) we will construct an object of

$$\begin{aligned}\mathsf{Eq}((\Pi z \in C)D(z), \lambda z.b(z), \lambda z.d(z)) \rightarrow&\\ \mathsf{Eq}(D(z), b(z), d(z))\ [I_\Pi, z \in C]&\end{aligned} \tag{6}$$

and we then get (4) by an application of the general lemma

$$(P \rightarrow Q) \rightarrow (\neg\neg P \rightarrow \neg\neg Q)$$

and $\rightarrow$-elimination. In order to prove (6) we introduce the new assumption

$$e \in \mathsf{Eq}((\Pi z \in C)D(z), \lambda z.b(z), \lambda z.d(z))$$

Since $\mathsf{apply}(f, z) \in D(z)\,[f \in (\Pi z \in C)D(z), z \in C]$, we get, by strong Eq–elimination and substitution,

$$\mathsf{apply}(\lambda z.b(z), z) = \mathsf{apply}(\lambda z.d(z), z) \in D(z)$$

which gives

$$b(z) = d(z) \in D(z)$$

under the assumptions

$$I_\Pi,\ z \in C,\ e \in \mathsf{Eq}((\Pi x \in C)D(x), \lambda x.b(x), \lambda x.d(x))$$

and by Eq-introduction and $\rightarrow$-introduction we get that $\lambda e.\mathsf{eq}$ is an object of (6).

□

The details of the full proof can be found in Salvesen [8].

4 Harrop formulas

We will define a class of stable formulas corresponding to Harrop-formulas in predicate logic [2]. The definition is made in two steps where the first is a straightforward translation of the definition of Harrop-formulas given for instance in [12] to type theory. The second step is made by reflection of the first, using the universe U. So first we define the class of Harrop-formulas by the following inductive definition.

- $\perp$ is a Harrop-formula.
- T is a Harrop-formula.
- If A is a type, $a \in A$ and $b \in A$ then $\mathsf{Eq}(A, a, b)$ is a Harrop-formula.
- If A and B are Harrop-formulas then $A \& B$ is a Harrop-formula.
- If A is a type and B is a Harrop-formula then $A \to B$ is a Harrop-formula.
- If A is a type and $B(x)$ is a Harrop-formula under the assumption that $x \in A$ then $(\forall x \in A)B(x)$ is a Harrop-formula.

Since $\neg A$ is identified with $A \to \perp$ we have that $\neg A$ is a Harrop-formula if A is a type. Note that if A is a Harrop-formula then A must be a type. So the notion of Harrop-formulas depends on the formulation of type theory.

We can now prove the following theorem.

Theorem 4 *If A is a Harrop-formula in ETT, then $\neg\neg A \to A$ is derivable in ETT.*

Proof. Straightforward by induction on the length of the proof that A is a Harrop-formula, using theorem 3.

$\square$

We shall extend the class of Harrop-formulas by reflection on the definition above, using the universe U. We first define the type H by

$$\mathsf{H} \equiv \{z \in \mathsf{U} \mid \mathsf{Type}(h(z))\}$$

where h is defined, using the recursion operator for U, so that

$$\begin{array}{rcl}
h(\widehat{\mathsf{T}}) & = & \widehat{\mathsf{T}} \in \mathsf{U} \\
h(\widehat{\perp}) & = & \widehat{\mathsf{T}} \in \mathsf{U} \\
h(\widehat{\mathsf{Eq}}(A, a, b) & = & \widehat{\mathsf{T}} \in \mathsf{U} \\
h(\widehat{\&}(A, B)) & = & \widehat{\&}(h(B), h(A)) \in \mathsf{U} \\
h(\widehat{\to}(A, B) & = & h(B) \in \mathsf{U} \\
h(\widehat{\forall}(A, B) & = & \widehat{\forall}(A, (x)h(B(x))) \in \mathsf{U} \\
h(A) & = & \widehat{\perp} \in \mathsf{U} \quad \text{all other canonical forms of } \mathsf{U}.
\end{array}$$

Using the induction principle of U and that h is recursively defined by reflections of stable propositions, we can prove that $\mathsf{Type}(h(x))$ is stable for all $x \in \mathsf{U}$:

$$(\forall x \in \mathsf{U})(\neg\neg\mathsf{Type}(h(x)) \to \mathsf{Type}(h(x))) \tag{1}$$

We can also prove that objects satisfying h are stable:

$$(\forall x \in \mathsf{U})(\mathsf{Type}(h(x)) \to (\neg\neg\mathsf{Type}(x) \to \mathsf{Type}(x))) \tag{2}$$

The proofs of (1) and (2) use the induction principle of U together with the definition of h.

Using (1), theorem 2 and (2) we obtain the following theorem.

Theorem 5 *The objects of* H *are stable, that is* $(\forall x \in \mathsf{H})(\neg\neg\mathsf{Type}(x) \to \mathsf{Type}(x))$.

In view of this theorem we may add a the clause

- if $a \in \mathsf{H}$ then $\mathsf{Type}(a)$ is a Harrop-formula.

to the inductive definition of Harrop formulas and still prove theorem 4.

A more detailed discussion of the class of Harrop-formulas can be found in Salvesen [8].

5 A counterexample

If $P(x) \equiv (\exists y \in B)Q(y)$ then a proof of $(*)$ would give us a method for constructing an object b of B such that $Q(b)$ holds from an object of $\{x \in A \mid (\exists y \in B)Q(y)\}$. We cannot expect this to hold in general since an object of $\{x \in A \mid (\exists y \in B)Q(y)\}$ does not contain any information of an object b of B for which $Q(b)$ holds. If we go outside the class of stable formulas, then we can actually find a counterexample to $(*)$:

Theorem 6 *If* $P(x) \equiv (\exists y \in \mathsf{N})T(x,x,y) \vee \neg(\exists y \in \mathsf{N})T(x,x,y)$, *where* T *is Kleene's T-predicate, and* $A \equiv \mathsf{N}$, *then* $(*)$ *cannot be derived in type theory extended with subsets.*

The idea of this counterexample comes from the refutation in Troelstra [11] of a generalization of the so called extended Church thesis, ECT_0:

$$\begin{aligned}&(\forall x)(P(x) \to (\exists y)Q(x,y)) \to \\ &\qquad (\exists u)(\forall x)(P(x) \to (\exists v)(T(u,x,v) \,\&\, Q(x,U(v))))\end{aligned} \tag{1}$$

where $P(x)$ must be an almost negative formula. Troelstra shows, by a diagonalization argument, that (1) is false if we put $P(x)$ equal to

$$(\exists y)T(x,x,y) \vee \neg(\exists y)T(x,x,y)$$

which is not almost negative. ECT_0 is closely related to $(*)$: By the same method as in the proof below, we can show, using $(*)$ and the axiom of choice in type theory, that there exists a term

$$h(x) \in \mathsf{N}\ [x \in \mathsf{N},\ y \in P(x)]$$

such that

$$(\forall x \in \mathsf{N})(P(x) \to (\exists y \in \mathsf{N})Q(x,y)) \ \to\ (\forall x \in \mathsf{N})(P(x) \to Q(x,h(x)))$$

Proof of theorem 6. Assume that we have a derivation of

$$(\forall x \in \{z \in \mathsf{N} \mid (\exists y \in \mathsf{N})T(z,z,y) \lor \neg(\exists y \in \mathsf{N})T(z,z,y)\}) \\ (\exists y \in \mathsf{N})T(x,x,y) \lor \neg(\exists y \in \mathsf{N})T(x,x,y) \tag{2}$$

Using the rules for subsets, it is easy to prove

$$(\forall x \in \{z \in A \mid P(z)\})Q(x) \ \&\ (\forall x \in A)(Q(x) \to R(x)) \ \to \\ (\forall x \in \{z \in A \mid P(z)\})R(x) \tag{3}$$

Since

$$(\forall x \in \mathsf{N})(((\exists y \in \mathsf{N})T(x,x,y) \lor \neg(\exists y \in \mathsf{N})T(x,x,y)) \ \to \\ (\exists z \in \mathsf{N})((z > 0 \to T(x,x,z \dot{-} 1)) \ \&\ (z = 0 \to \neg(\exists y \in \mathsf{N})T(x,x,y))))$$

is provable in first-order intuitionistic arithmetic, it must also be derivable in type theory. Hence, using (3), we obtain from (2)

$$(\forall x \in \{z \in \mathsf{N} \mid (\exists y \in \mathsf{N})T(z,z,y) \lor \neg(\exists y \in \mathsf{N})T(z,z,y)\}) \\ (\exists z \in \mathsf{N})((z > 0 \to T(x,x,z \dot{-} 1)) \ \&\ (z = 0 \to \neg(\exists y \in \mathsf{N})T(x,x,y))) \tag{4}$$

By the axiom of choice in type theory, we get from (4) a term h such that

$$h(x) \in \mathsf{N}\ [x \in \{z \in \mathsf{N} \mid (\exists y \in \mathsf{N})T(z,z,y) \lor \neg(\exists y \in \mathsf{N})T(z,z,y)\}] \tag{5}$$

and

$$(\forall x \in \{z \in \mathsf{N} \mid (\exists y \in \mathsf{N})T(z,z,y) \lor \neg(\exists y \in \mathsf{N})T(z,z,y)\}) \\ ((h(x) > 0 \to T(x,x,h(x) \dot{-} 1)) \& (h(x) = 0 \to \neg(\exists y \in \mathsf{N})T(x,x,y))) \tag{6}$$

Using subset-introduction, it easy to prove

$$(\forall x \in \{z \in A \mid P(z)\})Q(x) \ \to\ (\forall x \in A)(P(x) \to Q(x)) \tag{7}$$

From (6) and (7) we get

$$(\forall x \in \mathsf{N})((\exists y \in \mathsf{N})T(x,x,y) \lor \neg(\exists y \in \mathsf{N})T(x,x,y) \to \\ ((h(x) > 0 \to T(x,x,h(x) \dot{-} 1)) \ \& \\ (h(x) = 0 \to \neg(\exists y \in \mathsf{N})T(x,x,y)))) \tag{8}$$

From (5) we get, by subset introduction,

$$h(x) \in \mathsf{N}\ [x \in \mathsf{N},\ z \in ((\exists y{\in}\mathsf{N})T(x,x,y) \vee \neg(\exists y{\in}\mathsf{N})T(x,x,y))] \tag{9}$$

Since all functions formally derivable in type theory are mechanically computable, we know that the term $h(x)$ is extensionally equal to a partial recursive function. A tedious proof of this could be obtained by induction on the length of the derivation that the function has a type; informally we could just appeal to Church thesis. Let u_0 be a gödelnumber for this function. By soundness, we get from (9) that the proposition

$$\begin{aligned}(\forall x{\in}\mathsf{N})(((\exists y{\in}\mathsf{N})T(x,x,y) \vee \neg(\exists y{\in}\mathsf{N})T(x,x,y)) \ \rightarrow \\ (\exists w{\in}\mathsf{N})(T(u_0,x,w)\ \&\ U(w) = h(x)))\end{aligned}$$

is semantically true. Hence, we get from (8) that the following proposition is true:

$$\begin{aligned}(\forall x{\in}\mathsf{N})(((\exists y{\in}\mathsf{N})T(x,x,y) \vee \neg(\exists y{\in}\mathsf{N})T(x,x,y)) \ \rightarrow \\ ((\exists w{\in}\mathsf{N})T(u_0,x,w)\ \&(U(w) > 0 \rightarrow T(x,x,U(w)\dot{-}1))\ \& \\ (U(w) = 0 \rightarrow \neg(\exists y{\in}\mathsf{N})T(x,x,y)))) \end{aligned} \tag{10}$$

Following Troelstra [11] p.381 (or [12] p. 197), we can see that (10) is false and, hence, that (2) cannot be derived in type theory. Since

$$(\forall x{\in}\mathsf{N})\neg\neg((\exists y{\in}\mathsf{N})T(x,x,y) \vee \neg(\exists y{\in}\mathsf{N})T(x,x,y))$$

holds intuitionistically, we obtain from (10)

$$\begin{aligned}(\forall x{\in}\mathsf{N})\neg\neg((\exists w{\in}\mathsf{N})(T(u_0,x,w)\ \&(U(w) > 0 \rightarrow T(x,x,U(w)\dot{-}1))\ \& \\ (U(w) = 0 \rightarrow \neg(\exists y{\in}\mathsf{N})T(x,x,y)))) \end{aligned} \tag{11}$$

Let v_0 be such that $(\forall n \in \mathsf{N})((\exists w \in \mathsf{N})T(v_0,x,w) \leftrightarrow \{u_0\}(x) \simeq 0)$. From (11) we then obtain a contradiction:

$$(\exists w{\in}\mathsf{N})T(v_0,v_0,w) \ \leftrightarrow\ \{u_0\}(v_0) \simeq 0 \ \leftrightarrow\ \neg(\exists y{\in}\mathsf{N})T(v_0,v_0,y)$$

$\square$

Remark. Note that this proof shows that if we require our theory to be constructive, then we cannot expect $(*)$ to hold for all predicates $P(x)$. However, if we to ETT add the axiom

$$?_A \in A \vee (\neg A)$$

for each type A, as suggested in [9], then $(*)$ becomes provable for all predicates since the law of the excluded middle implies that all predicates are stable.

6 Conclusions

A straightforward introduction of subset types in type theory is problematic because the subset type is difficult to integrate with propositions as types. Although theorem 2 together with Harrop-formulas give some possibilities in ETT to retrieve information from a subset, it is not the kind of restrictions you want to have when using type theory.

It seems to us that if one wants to have a subset type which will work in practice, it must be possible to say that a proposition is true without explicitly showing a proof object. One way of obtaining this is to extend type theory with the two new forms of judgement $A\ \mathit{prop}$ and $A\ \mathit{true}$, meaning that A is a proposition and A is a true proposition, respectively. We would then no longer view the logical constants as abbreviations of the corresponding type theoretical constants, using the interpretation of propositions as types. The rules for predicate logic, with the quantifiers ranging over types, would then have to be added. The formation and introduction rules for subsets now become:

Subset-formation

$$\frac{A\ \mathit{type} \qquad B(x)\ \mathit{prop}\ [x \in A]}{\{x \in A \mid B(x)\}\ \mathit{type}}$$

Subset-introduction

$$\frac{a \in A \qquad B(a)\ \mathit{true}}{a \in \{x \in A \mid B(x)\}}$$

As for all the other types, there must now be two elimination rules:

Subset-elimination for types

$$\frac{a \in \{x \in A \mid B(x)\} \qquad c(x) \in C(x)\ [x \in A,\ B(x)\ \mathit{true}]}{c(a) \in C(a)}$$

Subset-elimination for propositions

$$\frac{a \in \{x \in A \mid B(x)\} \qquad C(x)\ \mathit{true}\ [x \in A,\ B(x)\ \mathit{true}]}{C(a)\ \mathit{true}}$$

In the first elimination rule we must have $C(x)\ \mathit{type}\ [x \in \{z \in A \mid B(z)\}]$ and in the second elimination rule $C(x)\ \mathit{prop}\ [x \in \{z \in A \mid B(z)\}]$.

It is now easy to derive $(*)$. By putting $B \equiv C \equiv P$ in subset-elimination for propositions, we get

$$P(x)\ \mathit{true}\ [x \in \{z \in A \mid P(z)\}]$$

from which we obtain $(*)$ by $\forall$-introduction.

Acknowledgements. We would like to thank Per Martin-Löf, Bengt Nordström and Kent Petersson for many discussions on adding subsets to type theory.

References

[1] The Prl Staff (R. Constable et al.). Implementing Mathematics with The Nuprl Proof Development System. *Prentice-Hall, 1986.*

[2] R. Harrop. Concerning formulas of the types $A \to B \vee C$, $A \to (Ex)B(x)$ in intuitionistic formal systems. *Journal of Symbolic Logic Vol. 25, No. 1, March 1960, pp. 27-32.*

[3] P. Martin-Löf. An intuitionistic theory of types: predicative part. In *Logic Colloquium '73, North-Holland, 1975.*

[4] P. Martin-Löf. Constructive Mathematics and Computer Programming. In *Sixth International Congress for Logic, Methodology, and Philosophy of Science, pp. 153-175. North-Holland, 1982.*

[5] P. Martin-Löf. Intuitionistic Type Theory. *Studies in Proof Theory, Lecture Notes, Bibliopolis, Napoli, 1984.*

[6] B. Nordström and K. Petersson. Types and specifications. In *Proceedings IFIP'83, Paris, pp. 915-920. Elsevier, Amsterdam 1983.*

[7] B. Nordström, K. Petersson and J.M. Smith. Programming in Martin-Löf's type theory. An introduction. *Monograph.* In preparation. To be published by *Oxford University Press.*

[8] A. Salvesen. Stable propositions in Martin-Löf's extensional type theory. In preparation.

[9] J.M. Smith. On a Nonconstructive Type Theory and Program Derivation. To appear in the proceedings of *Conference on Logic and its Applications, Bulgaria 1986 (Plenum Press).*

[10] W.W. Tait. Intensional interpretation of functionals of finite type I. *Journal of Symbolic Logic Vol. 32, No. 2, June 1967, pp. 198-212.*

[11] A.S. Troelstra. Notions of Realizability. In *Proceedings of the Second Scandinavian Logic Symposium, pp. 369-405. North-Holland, Amsterdam, 1971.*

[12] A.S. Troelstra. Metamathematical Investigation of Intuitionistic Arithmetic and Analysis. *Lecture Notes in Mathematics, No. 344, Springer-Verlag, 1973.*