The forget-restore principle: a paradigmatic example

Silvio Valentini
Dipartimento di Matematica Pura ed Applicata,
Università di Padova
Via Belzoni 7, I–35131 Padova, Italy
e-mail: silvio@math.unipd.it

1 Introduction

The aim of this note is to give a simple but instructive example of the forget-restore principle, conceived by Giovanni Sambin as a discipline for a constructive development of mathematics and first appeared in print in the introduction of Sambin and Valentini 1997. The best way to explain such a philosophical position is to quote from that paper: "*To build up an abstract concept from a row flow of data, one must disregard inessential details ... this is obtained by forgetting some information. To forget information is the same as to destroy something, in particular if there is no possibility of restoring that information ... our principle is that an abstraction is constructive ... when information ... is forgotten in such a way that it can be restored at will in any moment.*"

The example we want to show here refers to Martin-Löf's intuitionistic type theory (just *type theory* from now on). We assume knowledge of the main peculiarities of type theory, as formulated in Martin-Löf 1984 or Nordström *et al.* 1990.

Type theory is a logical calculus which adopts those notions and rules which keep *total* control of the amount of information contained in the different forms of judgment. However, type theory offers a way of "forgetting" information, that is, supposing A *set*, the form of judgment A *true*.

The meaning of A *true* is that there exists an element a such that $a \in A$ but it does not matter which particular element a is (see also the notion of proof irrelevance in de Bruijn 1980). Thus to pass from the judgment $a \in A$ to the judgment A *true* is a clear example of the forgetting process.

We will show that it is a constructive way to forget since, provided that there is a proof of the judgment A *true*, an element a such that $a \in A$ can be re-constructed.

Of course the simple solution of adding *only* the rule

$$\frac{a \in A}{A \ true}$$

allows to obtain such a result but is completely useless in practice. In fact, it does not allow to *operate* with judgments of the form $A\ true$ and, in our experience, judgments of this form are essential in developing constructive mathematics, like for instance in formal topology, and in developing metamathematical arguments (see for instance Sambin and Valentini 1997 and Maietti and Valentini 1997).

To obtain the same result, but avoiding this limitation, we provide a general calculus for expressions, directly inspired by Martin-Löf's Siena lectures in April 1983 (see Bossi and Valentini 1989). This calculus was first published in Valentini 1996 and is similar for instance to that in Nordström *et al.* 1990. The advantage of our calculus with respect to the other ones present in the literature is that its rules, besides to allow to express all of the judgments of basic type theory, also permit a rigorous treatment of judgments of the form $A\ true$.

2 The multi-level typed lambda calculus

The first idea for the definition of our calculus is to use a sort of simple typed λ-calculus (see Barendregt 1992). In this way it is possible both to abstract on variables and to preserve a decidable theory of equality, which is an essential feature to describe any logical system since decidability is necessary in order to recognize the correct application of the inference rules. On the other hand, to describe type theory a simple typed λ-calculus is not sufficient. Thus we define the following multi-level typed λ-calculus: the intuitive idea is to construct a *tower* of dependent typed λ-calculi, each one over another, marked by a *level*. Hence the rules of the multi-level typed λ-calculus are those of a simple typed λ-calculus enriched by a label which specifies the level.

$$\text{(assumption)} \qquad \frac{\Gamma \vdash N :_i M}{\Gamma, x :_{i-1} N \vdash x :_{i-1} N} \qquad i \geq 1$$

$$\text{(weakening)} \qquad \frac{\Gamma \vdash N :_i M \qquad \Gamma \vdash K :_j L}{\Gamma, x :_{i-1} N \vdash K :_j L} \qquad i \geq 1$$

$$\text{(abstraction)} \qquad \frac{\Gamma, x :_j N \vdash K :_i L}{\Gamma \vdash ((x :_j N)K) :_i ((x :_j N)L)}$$

$$\text{(application)} \qquad \frac{\Gamma \vdash N :_i ((x :_j L)M) \qquad \Gamma \vdash K :_j L}{\Gamma \vdash N(K) :_i M[x := K]}$$

The assumption rule tells that, if N is an expression of level greater than zero, then we may assume it to be inhabited. The weakening rule tells that we may add assumptions of the form $x :_{i-1} N$ provided that the level of N is greater than zero. Abstraction and application are like usual, except that they apply to any level; note that they do not change the level of an expression.

These rules by themselves are not sufficient to develop any logical calculus since no expression can be assigned a type because to prove the conclusion of a rule one should have already proved its premise(s). So, in order to start, one needs some axioms. The first thing one has to do is to settle the maximum level m needed to describe a particular theory; to this aim we will introduce the

symbol $*$ to indicate the only type of the highest level. One can then define all the other types downward from $*$ by means of axioms of the form $\vdash c :_m *$ for some constant c. Note that the only elements of $*$ are constants. Then, all the other axioms will have the form $\Gamma \vdash c :_{j-1} M$ for some constant c provided that $j > 0$ and there exists a type N such that $\Gamma \vdash M :_j N$. It is not difficult to recognize here some analogies with the approach to typed lambda-calculi used in the Pure Type Systems approach (see Barendregt 1992).

In the case of type theory, we define a chain

$$a :_0 A :_1 set :_2 *$$

to mean that a is an element of A which is a set, i.e. an element of *set*, which is the only element of $*$. Thus our first axiom is:

$$\vdash set :_2 *$$

We can now begin our description of type theory; to this aim we will follow the informal explanation by Martin-Löf in Martin-Löf 1984. We start by stating an axiom which introduces a constant for each set-constructor in correspondence with each formation rule of type theory. For instance, suppose we want to describe the set $\Pi(A, B)$; to this aim we add the axiom

$$\vdash \Pi :_1 (X :_1 set)(Y :_1 (x :_0 X)\ set)\ set$$

which means that Π is a set-constructor constant which gives a new set when applied to the set X and to the propositional function Y on elements of X. It is straightforward to verify that this is a correct axiom since

$$\vdash (X :_1 set)(Y :_1 (x :_0 X)\ set)\ set :_2 (X :_1 set)(Y :_1 (x :_0 X)\ set)\ *$$

The next step corresponds to the introduction rule(s): we add a new axiom for each kind of canonical element. Let us consider again the case of the set $\Pi(A, B)$; then we put

$$\vdash \lambda :_0 (X :_1 set)(Y :_1 (x :_0 X)\ set)(y :_0 (x :_0 X)\ Y(x))\ \Pi(X, Y)$$

which states that, if X is a set, Y is a propositional function on elements of X and y is a function which gives a proof of $Y(x)$ for any x in X, then $\lambda(X, Y, y)$ is an element of the set $\Pi(X, Y)$. Thus this axiom is just a rephrasing of the Π-introduction rule in Martin-Löf 1984.

Also the elimination rule becomes a new axiom; it defines the term-constructor constant introduced by the elimination rule. For instance for the set $\Pi(A, B)$, following the standard elimination rule (see the introduction of Martin-Löf 1984), we put

$$\begin{aligned}\vdash F :_0 (X :_1 set)(Y :_1 (x :_0 X)\ set)(Z :_1 (z :_0 \Pi(X, Y))\ set)\\ (c :_0 \Pi(X, Y))(d :_0 (y :_0 (x :_0 X)\ Y(x))\ Z(\lambda(X, Y, y)))\ Z(c)\end{aligned}$$

which states that, if X is a set, Y is a propositional function on elements of X, Z is a propositional function on elements of $\Pi(X, Y)$, c is an element of $\Pi(X, Y)$ and d is a method which maps any function y from x in X to $Y(x)$ into an element of $Z(\lambda(X, Y, y))$, then $F(X, Y, c, d)$ is an element of $Z(c)$.

In a similar way all of the rules of type theory become axioms of the multi-level typed λ-calculus.

The notion of level will not be necessary to prove the main theorem of this paper but it is useful to prove that the multi-level typed lambda-calculus is normalizing. In fact, because of the presence of the levels, the multi-level typed lambda-calculus is obtained just putting together many dependent typed lambda-calculi with constants which cannot interact one with the other. Hence one can adapt to this framework any normalization proof for a dependent typed lambda calculus present in the literature (cf. Capretta and Valentini 1997). Anyway, in order to simplify the notation, in the following we will not write all the indexes of the levels whenever they are not necessary.

3 The judgment $A\ true$

The main novelty of our approach with respect to a standard simple typed lambda calculus, besides the notion of level, is that, besides the judgments of the form $N : M$ together with their rules, we can introduce here also the new form of judgment "$M\ true$", whose intended meaning is that the type M is inhabited. The rules we require on this form of judgment are completely similar to those for the judgment $N : M$ in the previous section. This fact will be crucial in the proof of the next theorem 3.1 which links the judgments of the form $N : M$ with those of the form $M\ true$.

$$\text{(assumption)} \qquad \frac{\Gamma \vdash N :_i M}{\Gamma, x :_{i-1} N \vdash N\ true} \qquad i \geq 1$$

$$\text{(weakening)} \qquad \frac{\Gamma \vdash N :_i M \qquad \Gamma \vdash L\ true}{\Gamma, x :_{i-1} N \vdash L\ true} \qquad i \geq 1$$

$$\text{(abstraction)} \qquad \frac{\Gamma, x :_j N \vdash L\ true}{\Gamma \vdash ((x :_j N)L)\ true}$$

$$\text{(application)} \qquad \frac{\Gamma \vdash ((x :_j L)M)\ true \qquad \Gamma \vdash K :_j L}{\Gamma \vdash M[x := K]\ true}$$

It may be useful to note that in most of the previous rules, besides judgments of the form $M\ true$, it is necessary to use also those of the form $N : M$ and thus this calculus cannot be expressed independently from the previous one.

Like for the judgments of the form $N : M$ in the previous section, no type M can be proved to be inhabited, i.e. $\vdash M\ true$, unless some specific axioms are added. The intended meaning of the judgment $M\ true$ suggests to add the axiom $\Gamma \vdash M\ true$ whenever an axiom of the form $\Gamma \vdash c : M$ is present for some constant c. For instance, when we consider the set $\Pi(A, B)$ we will add

the following two axioms:

$$\vdash (X : set)(Y : (x : X)\ set)(y : (x : X)\ Y(x))\ \Pi(X, Y)\ true$$

which states that the type $(X : set)(Y : (x : X)\ set)(y : (x : X)\ Y(x))\ \Pi(X, Y)$ is inhabited; by using it, one can prove for instance that $\Gamma \vdash \Pi(A, B)\ true$, provided that $\Gamma \vdash A : set$ and $\Gamma, x : A \vdash B(x) : set$ and $\Gamma, x : A \vdash B(x)\ true$ hold, since if $\Gamma, x : A \vdash B(x)\ true$ holds then, by the next theorem 3.1, it is possible to construct an expression b such that $\Gamma, x : A \vdash b(x) : B(x)$;

$$\begin{array}{ll} \vdash & (X : set)(Y : (x : X)\ set)(Z : (z : \Pi(X, Y))\ set) \\ & (c : \Pi(X, Y))(d : (y : (x : X)\ Y(x))\ Z(\lambda(X, Y, y)))\ Z(c)\ true \end{array}$$

which shows $\Gamma \vdash C(c)\ true$ provided that $\Gamma \vdash A : set$, $\Gamma, x : A \vdash B(x) : set$, $\Gamma, z : \Pi(A, B) \vdash C(z) : set$, $\Gamma \vdash c : \Pi(A, B)$ and $\Gamma, y : (x : A)\ B(x) \vdash C(\lambda(A, B, y))\ true$ hold. Note that, if the set $C(z)$ does not depend on z, the last axiom can be simplified to obtain $\Gamma \vdash C\ true$ provided that $\Gamma \vdash A : set$, $\Gamma, x : A \vdash B(x) : set$, $\Gamma, z : \Pi(A, B) \vdash C : set$, $\Gamma \vdash \Pi(A, B)\ true$ and $\Gamma, y : (x : A)\ B(x) \vdash C\ true$ hold, since, by theorem 3.1, $\Gamma \vdash \Pi(A, B)\ true$ implies that there exists an element c such that $\Gamma \vdash c : \Pi(A, B)$.

Since the rules for the judgment $N : M$ are completely similar to those for the judgment $M\ true$ and whenever an axiom of the form $\Gamma \vdash c : M$ is added to the calculus also the axiom $\Gamma \vdash M\ true$ is added, we can prove the following theorem 3.1. It allows to give a formal counterpart of the intended meaning of the judgment $\Gamma \vdash M\ true$. Its proof, in one direction, shows how the reconstruct a witness for the judgment $M\ true$ while, in the other, it shows how it is possible to forget safely.

Theorem 3.1 *Let Σ be any set of axioms of the form $\Gamma \vdash c : K$, for some constant c and type K, and let Σ^* be obtained from Σ by suitably substituting some of the axioms $\Gamma \vdash c : K$ in Σ with the corresponding axiom $\Gamma \vdash K\ true$. Then $\Gamma \vdash M\ true$ is derivable from Σ^* if and only if there exists an expression N such that $\Gamma \vdash N : M$ is derivable from Σ.*

Proof In both directions the proof is by induction on the given proof. When we are "forgetting" we must start from below so that we know what can be forgotten. Let us show the inductive steps (provided Π is a proof, we will write Π^* to mean the proof obtained by inductive hypothesis).
(axiom)

$$\frac{\begin{array}{c}\Pi\\ \Gamma \vdash K :_i N\end{array}}{\Gamma \vdash c :_{i-1} K} \quad \Rightarrow \quad \frac{\begin{array}{c}\Pi\\ \Gamma \vdash K :_i N\end{array}}{\Gamma \vdash K\ true}$$

(assumption)

$$\frac{\begin{array}{c}\Pi\\ \Gamma \vdash N :_i M\end{array}}{\Gamma, x :_{i-1} N \vdash x :_{i-1} N} \quad \Rightarrow \quad \frac{\begin{array}{c}\Pi\\ \Gamma \vdash N :_i M\end{array}}{\Gamma, x :_{i-1} N \vdash N\ true}$$

(weakening)

$$\dfrac{\begin{array}{cc}\Pi & \Sigma\\ \Gamma \vdash N :_i M & \Gamma \vdash K :_j L\end{array}}{\Gamma, x :_{i-1} N \vdash K :_j L} \quad \Rightarrow \quad \dfrac{\begin{array}{cc}\Pi & \Sigma^*\\ \Gamma \vdash N :_i M & \Gamma \vdash L\ true\end{array}}{\Gamma, x :_{i-1} N \vdash L\ true}$$

(abstraction)

$$\dfrac{\begin{array}{c}\Pi\\ \Gamma, x :_i N \vdash K :_j L\end{array}}{\Gamma \vdash ((x :_i N)K) :_j ((x :_i N)L)} \quad \Rightarrow \quad \dfrac{\begin{array}{c}\Pi^*\\ \Gamma, x :_i N \vdash L\ true\end{array}}{\Gamma \vdash ((x :_i N)L)\ true}$$

(application)

$$\dfrac{\begin{array}{cc}\Pi & \Sigma\\ \Gamma \vdash N :_j ((x :_i L)M) & \Gamma \vdash K :_i L\end{array}}{\Gamma \vdash N(K) :_j M[x := K]} \quad \Rightarrow \quad \dfrac{\begin{array}{cc}\Pi^* & \Sigma\\ \Gamma \vdash ((x :_i L)M)\ true & \Gamma \vdash K :_i L\end{array}}{\Gamma \vdash M[x := K]\ true}$$

It should now be clear how we obtain the set of axioms Σ^* from the set of axioms Σ: we have to change only those axioms which appear on a modified proof and this is the reason why we have to "forget" from below: for instance in the rules of weakening or application only one the premises is modified and only the axioms on that premise have to be changed.

On the other side, in the case we are "restoring", we must start from above in such a way that an axiom (possibly in Σ^*) is replaced with a suitable instance of an axiom (in Σ).

(axiom)

$$\dfrac{\begin{array}{c}\Pi\\ \Gamma \vdash M :_i N\end{array}}{\Gamma \vdash M\ true} \quad \Rightarrow \quad \dfrac{\begin{array}{c}\Pi\\ \Gamma \vdash M :_i N\end{array}}{\Gamma \vdash c :_i M}$$

(assumption)

$$\dfrac{\begin{array}{c}\Pi\\ \Gamma \vdash N :_i M\end{array}}{\Gamma, x :_{i-1} N \vdash N\ true} \quad \Rightarrow \quad \dfrac{\begin{array}{c}\Pi\\ \Gamma \vdash N :_i M\end{array}}{\Gamma, x :_{i-1} N \vdash x :_{i-1} N}$$

(weakening)

$$\dfrac{\begin{array}{cc}\Pi & \Sigma\\ \Gamma \vdash N :_i M & \Gamma \vdash L\ true\end{array}}{\Gamma, x :_{i-1} N \vdash L\ true} \quad \Rightarrow \quad \dfrac{\begin{array}{cc}\Pi & \Sigma^*\\ \Gamma \vdash N :_i M & \Gamma \vdash K :_j L\end{array}}{\Gamma, x :_{i-1} N \vdash K :_j L}$$

(abstraction)

$$\dfrac{\begin{array}{c}\Pi\\ \Gamma, x :_i N \vdash L\ true\end{array}}{\Gamma \vdash ((x :_i N)L)\ true} \quad \Rightarrow \quad \dfrac{\begin{array}{c}\Pi^*\\ \Gamma, x :_i N \vdash K :_j L\end{array}}{\Gamma \vdash ((x :_i N)K) :_j ((x :_i N)L)}$$

(application)

$$\frac{\begin{array}{cc}\Pi & \Sigma\\ \Gamma \vdash ((x :_i L)M)\ true & \Gamma \vdash K :_i L\end{array}}{\Gamma \vdash M[x := K]\ true} \quad \Rightarrow \quad \frac{\begin{array}{cc}\Pi^* & \Sigma\\ \Gamma \vdash N :_j ((x :_i L)M) & \Gamma \vdash K :_i L\end{array}}{\Gamma \vdash N(K) :_j M[x := K]}$$

□

It is worth noting that in the process which transforms first the proof of $\Gamma \vdash N : M$ into $\Gamma \vdash M\ true$ and then into $\Gamma \vdash N' : M$ we will not in general obtain the same element, i.e. N and N' may differ for the constants used in the axioms with the same type.

4 Final remarks

What we have illustrated in the previous sections is just an example of the process of "forgetting"; for instance, as one of the referees of this paper has suggested, one could consider also the judgments $M\ type$ and $M\ element$ as a forgetting abbreviation for the judgment $M :_j N$ with $j > 0$ and $j = 0$ respectively and develop for these judgments a suitable calculus analogous to the one we proposed for the judgment $M\ true$.

Moreover it should be clear that what we have done is just a simple illustration of the forget-restore paradigma and that it is not a complete description of a full theory for judgments of the form $A\ true$ within type theory. In fact we chose to develop a dependent type multilevel lambda calculus since it is well suited for the framework of the Martin-Löf's dependent type theory that we have described but it is not of immediate application if we consider also the non-dependent part of the theory like for instance when we define $A \to B$ as $\Pi(A, (x : A)\ B)$ provided that the proposition B does not depend on the elements of A. For instance the rule

$$\frac{\Gamma \vdash A \to B\ true \qquad \Gamma \vdash A\ true}{\Gamma \vdash B\ true}$$

is admissible in our system but it is not derivable ; hence we have a too weak theory for judgments of the form $A\ true$. To solve this problem the first step is to be able to deal also with *assumptions* of the form $A\ true$, instead that only with those of the form $x : A$, when the variable x does not appear in the conclusion $B\ true$. This is not possible in a general dependent type calculus since even a conclusion of the form $B\ true$ may in general depend on the variables in the assumptions.

We can obtain this result if, when performing the forgetting process, we will take into account also which variables appear in the types in the conclusion of a rule. Thus we will have the following transformation of a *full* proof into a *forgetting* one:

$$\text{(assumption)} \quad \frac{\begin{array}{c}\Pi\\ \Gamma \vdash N :_i M\end{array}}{\Gamma, x :_{i-1} N \vdash x :_{i-1} N} \quad \Rightarrow \quad \frac{\begin{array}{c}\Pi\\ \Gamma \vdash N :_i M\end{array}}{\Gamma, N\ true \vdash N\ true}$$

since the variable x is introduced by the rule and hence cannot appear in N;

$$\text{(weakening)}\quad \frac{\begin{array}{cc}\Pi & \Sigma\\ \Gamma \vdash N :_i M & \Gamma \vdash K :_j L\end{array}}{\Gamma, x :_{i-1} N \vdash K :_j L} \quad \Rightarrow \quad \frac{\begin{array}{cc}\Pi & \Sigma^*\\ \Gamma \vdash N :_i M & \Gamma \vdash L\ true\end{array}}{\Gamma, N\ true \vdash L\ true}$$

since the variable x is assumed by weakening and hence it cannot appear in L. The case of the abstraction rule

$$\text{(abstraction)}\quad \frac{\begin{array}{c}\Pi\\ \Gamma, x :_i N \vdash K :_j L\end{array}}{\Gamma \vdash ((x :_i N)K) :_j ((x :_i N)L)}$$

deserves a more detailed analysis; in fact we can surely use the transformation that we have proposed in the proof of theorem 3.1, but, provided the variable x does not appear in L, also the following transformation can be used

$$\frac{\begin{array}{c}\Pi^*\\ \Gamma, N\ true \vdash L\ true\end{array}}{\Gamma \vdash ((N)L)\ true}$$

where we have introduced the new notation $((N)L)$ to mean that the abstracted variables does not appear in the body of the abstraction. Finally also for the application rule

$$\text{(application)}\quad \frac{\begin{array}{cc}\Pi & \Sigma\\ \Gamma \vdash N :_j ((x :_i L)M) & \Gamma \vdash K :_i L\end{array}}{\Gamma \vdash N(K) :_j M[x := K]}$$

two transformations are possible according to the variables which appear in the conclusion. The first is the one that we used in the proof of theorem 3.1 and it can be applied in any case. However, provided M does not depend on x, it is possible to use also the following

$$\frac{\begin{array}{cc}\Pi^* & \Sigma^*\\ \Gamma \vdash ((L)M)\ true & \Gamma \vdash L\ true\end{array}}{\Gamma \vdash M\ true}$$

It is now possible to change also the form of the axioms. Here we will show only a simple example. Suppose that we want to introduce the type $A \to B$. Then we need the following axioms:

$$\begin{array}{l}\vdash \to :_1 (X :_1 set)(Y :_1 set)\ set\\ \vdash \lambda :_0 (X :_1 set)(Y :_1 set)(y :_0 (x :_0 X)\ Y)\ X \to Y\\ \vdash ap :_0 (X :_1 set)(Y :_1 set)(f :_0 X \to Y)(x :_0 X)\ Y\end{array}$$

If we now consider the transformations used in the prove of theorem 3.1 we will obtain

$$\begin{array}{l}\vdash (X :_1 set)(Y :_1 set)(y :_0 (x :_0 X)\ Y)\ X \to Y\ true\\ \vdash (X :_1 set)(Y :_1 set)(f :_0 X \to Y)(x :_0 X)\ Y\ true\end{array}$$

but, provided that we use also the notation $((X)Y)$ for the abstractions when Y does not depends on the elements in X, we can add to them the following new axioms:

$$\vdash (X :_1 set)(Y :_1 set)(((X)\ Y)\ X \to Y)\ true$$
$$\vdash (X :_1 set)(Y :_1 set)(f :_0 X \to Y)((X)\ Y)\ true$$
$$\vdash (X :_1 set)(Y :_1 set)((X \to Y)(x :_0 X)\ Y)\ true$$
$$\vdash (X :_1 set)(Y :_1 set)((X \to Y)((X)\ Y))\ true$$

and it is straightforward to use the last one to derive the rule

$$\frac{\Gamma \vdash A \to B\ true \qquad \Gamma \vdash A\ true}{\Gamma \vdash B\ true}$$

Since any of the new axioms is the result of a forgetting process from a standard axiom and we can restore it simply by adding the abstracted variables, which can be done in an algorithmic way, this is again an instance of a constructive way of forgetting and a theorem like theorem 3.1 can be proved also in this case.

Bibliography

Barendregt, H. P. (1992). *Lambda-calculi with types*, in "Handbook of logic and computer science", vol. II, S. Abramski, D. M. Gabbay and T. S. Maibaum, eds., vol. 2, pp. 118-309, Oxford University Press.

Bossi, A. and Valentini, S. (1989). *The expressions with arity*, internal report, Dip. Scienze dell'Informazione, Univ. Milano N. 61/89.

Capretta, V. and Valentini, S. (1997). *A general method to prove the normalization theorem for first and second order typed λ-calculi*, to appear.

de Bruijn, N. G. (1980). *A survey of the project Automath.*, in "To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism", J. P. Seldin and J. R. Hyndley, eds., Academic Press, London, pp. 589-606.

Martin-Löf, P. (1984). "Intuitionistic type theory", notes by Giovanni Sambin of a series of lectures given in Padua, June 1980, Bibliopolis, Neaples.

Maietti, M. E. and Valentini, S. (1997). *Why you should not add power-set to Martin-Löf intuitionistic set theory* to appear.

Nordström, B., Petersson, K. and Smith, J.M. (1990). "Programming in Martin-Löf's Type Theory, an introduction", Clarendon Press, Oxford.

Sambin, G. and Valentini, S. (1997). *Building up a toolbox for Martin-Löf's type theory.*, in "Twenty five years of Constructive Type Theory", Venice.

Valentini, S. (1996) *Another introduction to Martin-Löf's Type Theory*, in "Trends in Theoretical Informatics", R. Albrecht and H. Herre, eds., Schriftenreihe der Österreichischen Computer Gesellscaft, Bd. 89, München.