Inductive Sets and Families in Martin-Löf's Type Theory and Their Set-Theoretic Semantics

Peter Dybjer

Chalmers University of Technology

Abstract

Martin-Löf's type theory is presented in several steps. The kernel is a dependently typed λ-calculus. Then there are schemata for inductive sets and families of sets and for primitive recursive functions and families of functions. Finally, there are set formers (generic polymorphism) and universes. At each step syntax, inference rules, and set-theoretic semantics are given.

1 Introduction

Usually Martin-Löf's type theory is presented as a closed system with rules for a finite collection of set formers. But it is also often pointed out that the system is in principle open to extension: we may introduce new sets when there is a need for them. The principle is that a set is by definition inductively generated - it is defined by its introduction rules, which are rules for generating its elements. The elimination rule is determined by the introduction rules and expresses definition by primitive recursion on the way the elements of the set are generated. (In this paper I shall use the term primitive recursive for the kind of recursion you have in type theory, which includes primitive recursive functionals and 'structural' recursion on an arbitrary inductive (=inductively defined) set (or family) including transfinite recursion.)

Backhouse [3] et.al. [4] exhibited a schema for *inductive sets* which delimits a class of definitions admissible in Martin-Löf's type theory which includes all the standard operations for forming small sets except the equality set. This schema extends Schroeder-Heister's schema for the logical constants [13, 14] to the type-theoretic case, where proof objects are explicitly represented in the theory.

Coquand and Paulin [5] and Dybjer [7] extended Backhouse's schema to incorporate *inductive families* and thus also *inductive predicates*. (Coquand and Paulin [5] presented their schema as an extension of impredicative higher order logic and the calculus of constructions, but the formal pattern is much the same as the one in Dybjer [7].) This schema covers all the standard operations for forming small sets including the equality set. It also subsumes Martin-Löf's schema for inductive predicates in predicate logic [8].

In this paper I give a somewhat different presentation of the schema. One difference is that also definitions of functions by primitive recursion are presented schematically (much like in Martin-Löf [9]) rather than by the usual kind of elimination rules. I also separate the presentation of the process of inductive generation of sets and families from the process of introducing parameters (generic polymorphism). Moreover, I present a version of type theory without a logical framework

[1]This is a slightly modified version of a paper with the same title which appeared in the Proceedings of the First Workshop on Logical Frameworks, Antibes, May 1990. Editors G. Huet and G. Plotkin. The research was partly supported by ESPRIT Basic Research Action "Logical Frameworks" and Styrelsen för Teknisk Utveckling.

and without an underlying theory of expressions (like in Martin-Löf's presentations of type theory before and including the book [10]).

I also show how to interpret type theory, with the schema, in classical set theory. This gives a non-intended but useful interpretation, compare Troelstra [16, page 2]: 'The simplest interpretation of $\mathbf{ML_0}$ is in terms of a hierarchy within classical set theory, where Π, Σ, etc, correspond to the formation of cartesian products, disjoint unions etc. as already indicated above; function, i.e., elements of cartesian products are regarded as equal if for each argument their values are equal, etc.'

Salvesen [12] presented details of such an interpretation of type theory. Well-orderings and the first universe were interpreted as inductively defined sets obtained by iterations of continuous operators.

Coquand and Paulin [5] proposed to give a set-theoretic interpretation of their schema for *inductive sets* by (i) translating the introduction rules defining a set to a strictly positive set operator in type theory; (ii) introducing rules for fixed points of such set operators in type theory and showing that the corresponding rules of the schema can be derived; (iii) interpreting type-theoretic strictly positive operators as ω_n-continuous functors on the category of sets (assuming a theory without universes).

In this paper I use Aczel's [1] notion of rule set rather than continuous functors. It is really only a variation, since a rule set generates a continuous operator. But it allows a direct concrete translation of the type-theoretic introduction rules to set-theoretic rule sets and generalizes the concrete construction of the term algebra T_Σ on a first order signature Σ.

I also interpret *inductive families*.

Even though the interpretation is a non-intended one, there is an analogy with Martin-Löf's intuitive justifications of the rules of type theory, whereby the formation rule receives its meaning from the introduction rules and the elimination rule receives its meaning from the equality rules.

I would also like to mention that Aczel [2] has shown how to interpret certain inductive sets, such as the well-orderings of type theory, in a *constructive set theory* (which itself can be interpreted in Martin-Löf's type theory). Does it follow that the whole of Martin-Löf's type theory can be interpreted in this constructive set theory?

Type theory is presented in the following steps.

- The dependently typed λ-calculus (section 2). This is like the simply typed λ-calculus with Π instead of $\rightarrow$. It consists essentially of the general rules and the rules for Π in Martin-Löf [10] except *eta*-conversion. (In the present *intensional* version of type theory of Martin-Löf 1986 there is no *eta*-conversion on the level of *sets* but only on the level of *types*.)

- Schema for inductive sets (section 3). This part of the schema is closely related to the well-orderings.

- Schema for primitive recursive function definitions (section 4).

- Schema for inductive families (section 5). This generalizes the schema for inductive sets. The simpler case is presented separately for the purpose of the presentation only.

- Schema for primitive recursive families of functions (section 6). This generalizes the schema for primitive recursive functions.

- Generic set formers. The fact that a definition may depend on parameters gives rise to set formers or generic polymorphism (section 7). Typical ambiguity is also discussed briefly, and it is noted that the interpretation allows polymorphic constructors, but not polymorphic recursive functions. Moreover, the possibilty of internalizing the schema is discussed.
- Universes (section 8).

At each step I first give syntax, then inference rules, and finally a set-theoretic interpretation.

In this paper I don't discuss simultaneous induction and recursion. The reader is referred to Dybjer [7] for this and also for some examples of what can be defined using the schema.

2 The dependently typed λ-calculus

We use ordinary notation, but omit mentioning variable restrictions, etc.

2.1 Expressions

Set expressions:

$$A \quad ::= \quad \Pi x : A_0.A_1[x].$$

Element expressions:

$$a \quad ::= \quad x \mid \lambda x : A.a[x] \mid a_1(a_0).$$

Context expressions:

$$? \quad ::= \quad \epsilon \mid ?, x : A.$$

Judgement expressions:

$$J \quad ::= \quad ? \ \mathit{context} \mid ? \vdash A \ \mathit{set} \mid ? \vdash a : A \mid ? \vdash A = A' \mid ? \vdash a = a' : A.$$

2.2 Inference rules

Some premises are omitted.

General rules:

$$\epsilon \ \mathit{context} \qquad \frac{? \ \mathit{context} \qquad ? \vdash A \ \mathit{set}}{?, x : A \ \mathit{context}}$$

$$\frac{? \vdash A \ \mathit{set}}{? \vdash A = A} \qquad \frac{? \vdash a : A}{? \vdash a = a : A}$$

$$\frac{? \vdash A = A'}{? \vdash A' = A} \qquad \frac{? \vdash a = a' : A}{? \vdash a' = a : A}$$

$$\frac{? \vdash A = A' \qquad ? \vdash A' = A''}{? \vdash A = A''} \qquad \frac{? \vdash a = a' : A \qquad ? \vdash a = a'' : A}{? \vdash a = a'' : A}$$

$$\frac{? \vdash A = A' \qquad ? \vdash a : A}{? \vdash a : A'} \qquad \frac{? \vdash A = A' \qquad ? \vdash a = a' : A}{? \vdash a = a' : A'}$$

$$\frac{? \vdash A\ set}{?, x : A \vdash x : A}$$

$$\frac{? \vdash A_0\ set \qquad ? \vdash A_1\ set}{?, x : A_0 \vdash A_1\ set} \qquad \frac{? \vdash A_0\ set \qquad ? \vdash a : A_1}{?, x : A_0 \vdash a : A_1}$$

Rules for the cartesian product of a family of sets:

$$\frac{? \vdash A_0\ set \qquad ?, x : A_0 \vdash A_1[x]\ set}{? \vdash \Pi x : A_0.A_1[x]\ set} \qquad \frac{? \vdash A_0 = A_0' \qquad ?, x : A_0 \vdash A_1[x] = A_1'[x]}{? \vdash \Pi x : A_0.A_1[x] = \Pi x : A_0'.A_1'[x]}$$

$$\frac{?, x : A_0 \vdash a[x] : A_1}{? \vdash \lambda x : A_0.a[x] : \Pi x : A_0.A_1[x]} \qquad \frac{?, x : A_0 \vdash a[x] = a'[x] : A_1}{? \vdash \lambda x : A_0.a[x] = \lambda x : A_0.a'[x] : \Pi x : A_0.A_1[x]}$$

$$\frac{? \vdash a_1 : \Pi x : A_0.A_1[x] \qquad ? \vdash a_0 : A_0}{? \vdash a_1(a_0) : A_1[a_0]} \qquad \frac{? \vdash a_1 = a_1' : \Pi x : A_0.A_1[x] \qquad ? \vdash a_0 = a_0' : A_0}{? \vdash a_1(a_0) = a_1'(a_0') : A_1[a_0]}$$

$$\frac{?, x : A_0 \vdash a_1[x] : A_1[x] \qquad ? \vdash a_0 : A_0}{? \vdash (\lambda x : A_0.a_1[x])(a_0) = a_1[a_0] : A_1[a_0]}$$

2.3 Interpretation of expressions

The basic idea of the interpretation is to interpret a type-theoretic concept as the corresponding set-theoretic concept, which usually has the same name. So a (type-theoretic) set is interpreted as a (set-theoretic) set, an element of a set as an element of a set, (definitional) equality as extensional equality, (type-theoretic) cartesian product as (set-theoretic) cartesian product, function as function graph, etc. A context is interpreted as a set of assignments.

Let $[\![a]\!]\rho$ be the denotation of the expression a under the assignment ρ. This assigns a set to each variable in a finite list of variables which includes all variables which are free in a. Let $\emptyset$ be the empty assignment and let ρ_x^u abbreviate $\rho \cup \{\langle x, u\rangle\}$. Let also $[\![a]\!]$ abbreviate $[\![a]\!]\emptyset$.

The interpretation function is partial. Partiality is introduced in the interpretation of application. But the interpretation of a derivable judgement will be defined and true.

The method with a partial interpretation function has also been used by Streicher for a categorical interpretation of the calculus of constructions [15].

Interpretation of set expressions:

$$[\![\Pi x : A_0.A_1[x]]\!]\rho \quad = \quad \prod_{u \in [\![A_0]\!]\rho} [\![A_1[x]]\!]\rho_x^u.$$

This is defined iff $[\![A_0]\!]\rho$ is defined and $[\![A_1[x]]\!]\rho_x^u$ is defined whenever $u \in [\![A_0]\!]\rho$.

Interpretation of element expressions:

$$[\![x]\!]\rho \quad = \quad \rho(x).$$

This is always defined.

$$[\![\lambda x : A.a[x]]\!]\rho \quad = \quad \{\langle u, [\![a[x]]\!]\rho_x^u\rangle | u \in [\![A]\!]\rho\}.$$

This is defined iff $[\![A]\!]\rho$ is defined and $[\![a[x]]\!]\rho_x^u$ is defined whenever $u \in [\![A]\!]\rho$.

$$[\![a_1(a_0)]\!]\rho \quad = \quad ([\![a_1]\!]\rho)([\![a_0]\!]\rho).$$

This is defined iff $[\![a_1]\!]\rho$ and $[\![a_0]\!]\rho$ are defined, and $[\![a_1]\!]\rho$ is a function the domain of which contains $[\![a_0]\!]\rho$. (Observe that it is possible to interpret polymorphic application in set theory. This is not the case for interpretations of type theory in general, compare Streicher [15].)

Interpretation of context expressions:

$$[\![\epsilon]\!] \quad = \quad \{\emptyset\}.$$

This is always defined.

$$[\![?\,, x : A]\!] \quad = \quad \{\rho^u_x | \rho \in [\![?]\!] \wedge u \in [\![A]\!]\rho\}.$$

This is defined iff $?$ is defined and $[\![A]\!]\rho$ is defined whenever $\rho \in [\![?]\!]$.

Interpretation of judgement expressions:

$$[\![?\ \mathit{context}]\!] \quad \text{iff} \quad [\![?]\!] \text{ is a set of assignments.}$$

This is defined iff $[\![?]\!]$ is defined.

$$[\![? \vdash A\ \mathit{set}]\!] \quad \text{iff} \quad [\![A]\!]\rho \text{ is a set whenever } \rho \in [\![?]\!].$$

This is defined iff $[\![?]\!]$ is defined and if $[\![A]\!]\rho$ is defined whenever $\rho \in [\![?]\!]$.

$$[\![? \vdash a : A]\!] \quad \text{iff} \quad [\![a]\!]\rho \in [\![A]\!]\rho \text{ whenever } \rho \in [\![?]\!].$$

This is defined iff $[\![?]\!]$ is defined and if $[\![a]\!]\rho$ and $[\![A]\!]\rho$ are defined whenever $\rho \in [\![?]\!]$.

$$[\![? \vdash A = A']\!] \quad \text{iff} \quad [\![A]\!]\rho = [\![A']\!]\rho \text{ whenever } \rho \in [\![?]\!].$$

This is defined iff $[\![?]\!]$ is defined and if $[\![A]\!]\rho$ and $[\![A']\!]\rho$ are defined whenever $\rho \in [\![?]\!]$.

$$[\![? \vdash a = a' : A]\!] \quad \text{iff} \quad [\![a]\!]\rho = [\![a']\!]\rho \wedge [\![a]\!]\rho \in [\![A]\!]\rho \wedge [\![a']\!]\rho \in [\![A]\!]\rho \text{ whenever } \rho \in [\![?]\!].$$

This is defined iff $[\![?]\!]$ is defined and if $[\![a]\!]\rho$, $[\![a']\!]\rho$, and $[\![A]\!]\rho$ are defined whenever $\rho \in [\![?]\!]$.

2.4 Soundness of the inference rules

Checking the soundness of the inference rules means checking that the interpretation of the conclusion of a rule is defined and true whenever the interpretation of the premises are defined and true.

It is quite straightforward to check the soundness of all the inference rules. As an illustration we show the soundness of the rule of application. The premises are interpreted as

$$[\![a_1]\!]\rho \in \prod_{u \in [\![A_0]\!]\rho} [\![A_1[x]]\!]\rho^u_x \text{ whenever } \rho \in [\![?]\!]$$

and

$$[\![a_0]\!]\rho \in [\![A_0]\!]\rho \text{ whenever } \rho \in [\![?]\!].$$

From this we conclude that

$$([\![a_1]\!]\rho)([\![a_0]\!]\rho) \in [\![A_1[x]]\!]\rho^{[\![a_0]\!]\rho}_x \text{ whenever } \rho \in [\![?]\!],$$

and hence the conclusion of the rule follows, since

$$[\![A_1[x]]\!]\rho^{[\![a_0]\!]\rho}_x = [\![A_1[a_0]]\!]\rho.$$

2.5 Telescopes

In the description of the schema below we shall frequently refer to sequences of dependent sets, to sequences (tuples) of elements, and to sequences of typings of elements. De Bruijn has introduced the term *telescope* for such sequences of dependent sets. Telescopes are closely related to contexts, they are so as to speak contexts treated as objects. Telescopes can also be viewed as obtained by iterating the Σ-construction.

It is intended that the reader view the terms telescope, tuple, etc., and certain associated notations as abbreviations and reduce them to formal notions of type theory in a way to be suggested below. (The description is not complete, and sometimes the notation needs to be interpreted with some good will in order to make sense.)

The new notation is explained as follows:

- As is a telescope means that A_1 set, $A_2[x_1]$ set $(x_1 : A_1), \ldots, A_n[x_1, \ldots, x_{n-1}]$ $(x_1 : A_1, \ldots, x_{n-1} : A_{n-1})$;
- $As = As'$ means that $A_1 = A_1$ set, $A_2[x_1] = A_2[x_1]$ set $(x_1 : A_1), \ldots, A_n[x_1, \ldots, x_{n-1}] = A'_n[x_1, \ldots, x_{n-1}]$ $(x_1 : A_1, \ldots, x_{n-1} : A_{n-1})$;
- $as :: As$ means that $a_1 : A_1, a_2 : A_2[a_1], \ldots, a_n : A_n[a_1, \ldots, a_{n-1}]$;
- $as = as' :: As$ means that $a_1 = a'_1 : A_1, a_2 = a'_2 : A_2[a_1], \ldots, a_n = a'_n : A_n[a_1, \ldots, a_{n-1}]$

respectively.

We also write $f(as)$ for $f(a_1, \ldots, a_n)$, $f(as, bs)$ for $f(a_1, \ldots, a_n, b_1, \ldots, b_m)$, etc.

An alternative approach would be to extend type theory with *formal* notions of telescopes and tuples. In addition to the standard forms of judgement we would have the new forms As is a telescope; $As = As'$; $as :: As$; $as = as' :: As$. As other forms of judgement these judgements would be made under assumptions. We would then have suitable rules for forming telescopes and tuples. Furthermore, if we have telescopes, a context can be viewed as a single assumption $xs :: As$. Compare also the discussion in section 7 on internalization of the schema.

The set-theoretic semantics can be extended to telescopes and tuples. When we write $us \in [\![As]\!]$, we understand that us is a tuple $\langle u_1, \ldots, u_n \rangle$ such that $u_1 \in [\![A_1]\!], \ldots, u_n \in [\![A_n[x_1, \ldots, x_{n-1}]]\!]^{u_1 \ldots u_{n-1}}_{x_1 \ldots x_{n-1}}$.

We also use index notation such as $(a_k)_k$, $(A_k)_k$, and $(a_k : A_k)_k$ to stand for $a_1, \ldots, a_n$, $A_1, \ldots, A_n$, and $a_1 : A_1, \ldots, a_n : A_n$ respectively. This will be used, for example, to talk about *non-dependent* telescopes of the form $(A_k)_k$.

3 Schema for inductive sets

We have now presented the syntax and rules of the dependently typed λ-calculus. Call this theory T_0. T_0 can be extended successively obtaining the theories $T_1, T_2, \ldots$.

There are two different kinds of extensions.

The first kind is when T_{n+1} is obtained from $T = T_n$ by adding formation and introduction rules for a new set former P (see section 3, 5, and 7).

The second kind is when T_{n+1} is obtained from $T = T_n$ by adding a new function constant f, which is defined by primitive recursion on some set (or family) and is specified by its type and its

computation rules (see section 4, 6, and 7). In this way we get schematic elimination and equality rules.

We first treat the simple case without parameters and inductive families.

3.1 Expressions

Set expressions:

$$A ::= P.$$

Element expressions

$$a ::= intro_i(as, (b_k)_k).$$

3.2 Inference rules

(J abbreviates $? \vdash J$.)
Formation rules:

$$P\ set,$$

$$P = P.$$

The ith *introduction rules*:

$$\frac{as :: Gs_i \qquad (b_k : Hs_{ik}[as] \to P)_k}{intro_i(as, (b_k)_k) : P},$$

$$\frac{as = as' :: Gs_i \qquad (b_k = b'_k : Hs_{ik}[as] \to P)_k}{intro_i(as, (b_k)_k) = intro_i(as', (b'_k)_k) : P},$$

where

- Gs_i is a telescope relative to T;
- $Hs_{ik}[xs]$ is a telescope relative to T in the context $xs :: Gs_i$ for each k.

3.3 Inductive sets in set theory

We shall use Aczel's [1] set-theoretic notion of rule set to interpret the introduction rules for a new set former. The set defined inductively by a rule set is the least set closed under all rules in the rule set.

A *rule* on a base set U in Aczel's sense is a pair of sets $\langle u, v\rangle$, often written

$$\frac{u}{v},$$

such that $u \subseteq U$ and $v \in U$.

Let Φ be a set of rules on U.

A set w is Φ-*closed* if

$$\frac{u}{v} \in \Phi \wedge u \subseteq w \supset v \in w.$$

There is a least Φ-closed set

$$\mathcal{I}(\Phi) = \bigcap\{w \subseteq U | w\ \Phi - closed\},$$

the set inductively defined by Φ.

3.4 Interpretation of expressions

Interpretation of set expressions:

$$[\![P]\!]\rho \quad = \quad \mathcal{I}(\Phi_P),$$

where

$$\Phi_P = \bigcup_i \{\frac{\bigcup_k \mathrm{ran}\ v_k}{\langle |intro_i|, us, (v_k)_k\rangle} | us \in [\![Gs_i]\!], (v_k \in [\![Hs_{ik}[xs]]\!]^{us}_{xs} \to U)_k\},$$

where $|intro_i| \in \omega$ is a code for the constructor $intro_i$, and $U = V_\alpha$, the set of sets generated before stage α in the cumulative hierarchy, where α is chosen so that Φ_P is a rule set on U. This induces the following requirements on the ordinal α:

- V_α is closed under tupling, that is, α is a limit ordinal;
- $\omega \subseteq V_\alpha$, that is, $\omega \leq \alpha$;
- $[\![Gs_i]\!] \subseteq V_\alpha$ for all i;
- $[\![Hs_{ik}[xs]]\!]^{us}_{xs} \to V_\alpha \subseteq V_\alpha$ for all $us \in [\![Gs_i]\!]$ and all ik. This can be achieved if $[\![Hs_{ik}[xs]]\!]^{us}_{xs} \subseteq V_\alpha$ and if $\mathrm{card}\,[\![Hs_{ik}[xs]]\!]^{us}_{xs} < \mathrm{card}\,\alpha$ for all $us \in [\![Gs_i]\!]$ and all ik. Because assume that $v_k \in [\![Hs_{ik}[xs]]\!]^{us}_{xs} \to V_\alpha$. Then for each $ws \in [\![Hs_{ik}[xs]]\!]^{us}_{xs}$ we have $\langle ws, v_k(ws)\rangle \in V_{\beta_{ws}}$ for some ordinal $\beta_{ws} < \alpha$. Let $\beta = \sup_{ws \in [\![Hs_{ik}[xs]]\!]^{us}_{xs}} \beta_{ws} < \alpha$. Hence, $v_k = \{\langle ws, v_k(ws)\rangle | ws \in [\![Hs_{ik}[xs]]\!]^{us}_{xs}\} \subseteq V_\beta$. So $v_k \in V_\alpha$.

(If the theory T does not include universes, then we can find $\alpha < \omega_\omega$ (assuming $\omega_{\alpha+1} = 2^{\omega_\alpha}$), because the rank and cardinality of the interpretation of sets constructed without universes are $< \omega_\omega$.)

Interpretation of element expressions:

$$[\![intro_i(as, (b_k)_k)]\!]\rho \quad = \quad \langle |intro_i|, [\![as]\!]\rho, ([\![b_k]\!]\rho)_k\rangle.$$

3.5 Soundness of the inference rules

Formation rule. We have already shown that $[\![P]\!]\rho$ is a set.

The introduction rule is sound because assume that $us \in [\![Gs_i]\!]\rho$ whenever $\rho \in [\![?]\!]$, and $v_k \in [\![Hs_{ik}[xs]]\!]\rho^{us}_{xs} \to [\![P]\!]$, whenever $\rho \in [\![?]\!]$, for all k. Then it follows that $\bigcup_k \mathrm{ran}\ v_k \subseteq [\![P]\!]$ and hence since $[\![P]\!]$ is Φ_P-closed $\langle |intro_i|, us, (v_k)_k\rangle \in [\![P]\!]$.

3.6 Logical consistency

Absurdity $(-)$ is interpreted as the empty set, the set which is defined by the empty list of introduction rules. We get that $[\![-]\!] = \emptyset$, so we cannot have $a : -$, since this entails $[\![a]\!] \in \emptyset$. Hence the set-theoretic interpretation shows the logical consistency of Martin-Löf's type theory.

4 Primitive recursive functions

Functions can be defined by recursion on the way the elements of P are generated (primitive or structural recursion). Here we give a schema for such definitions rather than a single elimination rule.

4.1 Syntax

$$a ::= f.$$

4.2 Inference rules

Elimination rules:

$$f : \Pi z : P.C[z],$$
$$f = f : \Pi z : P.C[z].$$

The ith *equality rules*:

$$\frac{as :: Gs_i \qquad (b_k : Hs_{ik}[as] \to P)_k}{f(intro_i(as, (b_k)_k)) = d_i(as, (b_k, \lambda zs :: Hs_{ik}[as].f(b_k(zs)))_k) : C[intro_i(as, (b_k)_k)]},$$

where

- $C[z]$ is a set in the context $z : P$;
- $$\begin{aligned} d_i \quad : \quad & \Pi xs :: Gs_i. \\ & (\Pi y_k : (Hs_{ik}[xs] \to P). \\ & \ \Pi y'_k : (\Pi zs :: Hs_{ik}[xs].C[y_k(zs)]))_k. \\ & C[intro_i(xs, (y_k)_k)]. \end{aligned}$$

4.3 Primitive recursive functions in set theory

A rule set Φ is *deterministic* if

$$\frac{u}{v} \in \Phi \wedge \frac{u'}{v} \in \Phi \supset u = u'.$$

If Φ is deterministic, functions on $\mathcal{I}(\Phi)$ can be defined by recursion on the way the elements in $\mathcal{I}(\Phi)$ are generated.

4.4 Interpretation of expressions

Since Φ_P is deterministic, type-theoretic primitive recursion can be interpreted as set-theoretic primitive recursion on $\mathcal{I}(\Phi_P)$. Let

$$[\![f]\!]\rho \quad = \quad \mathcal{I}(\Psi_f),$$

where

$$\begin{aligned} \Psi_f \quad = \quad & \bigcup_i \{ \frac{\bigcup_k \{\langle v_k(ws), v'_k(ws)\rangle | ws \in [\![Hs_{ik}[xs]]\!]^{us}_{xs}\}}{\langle\langle |intro_i|, us, (v_k)_k\rangle, [\![d_i]\!](us, (v_k, v'_k)_k)\rangle} | \\ & us \in [\![Gs_i]\!], \\ & (v_k \in [\![Hs_{ik}[xs]]\!]^{us}_{xs} \to [\![P]\!], \\ & \ v'_k \in \prod_{ws \in [\![Hs_{ik}[xs]]\!]^{us}_{xs}} [\![C[y_k(zs)]]\!]^{ws\ v_k}_{zs\ y_k})_k \}. \end{aligned}$$

Ψ_f is a rule set on $\sum_{w \in [\![P]\!]} [\![C[z]]\!]^w_z$. This is easily proved by Φ_P-induction since the pairs in $\mathcal{I}(\Psi_f)$ are generated in parallel with the elements of $\mathcal{I}(\Phi_P)$ and since the requirements on d_i ensure that the second component is in $[\![C[z]]\!]^w_z$.

4.5 Soundness of the inference rules

Since Φ_P is deterministic Ψ_f defines a function on $[\![P]\!]$. Hence

$$[\![f]\!]\rho \in \prod_{w \in [\![P]\!]\rho} [\![C[z]]\!]\rho^w_z,$$

and the elimination rule is validated.

To prove the soundness of the equality rules we need to prove three things: two memberships and an equality. The memberships are immediate. The equality is a direct consequence of the definition of Ψ_f.

5 Inductive families

We now treat the more general case of inductively defined families of sets, but still postpone the discussion of parameters.

5.1 Expressions

Set expressions:

$$A ::= P(as).$$

Element expressions

$$a ::= intro_i(as, (b_k)_k).$$

5.2 Inference rules

Formation rules:

$$\frac{as :: Is}{P(as)\ set},$$

$$\frac{as :: Is}{P(as) = P(as)},$$

where

- Is is a telescope relative to T.

The ith *introduction rules*:

$$\frac{as :: Gs_i \qquad (b_k : \Pi zs :: Hs_{ik}[as].P(qs_{ik}[as, zs]))_k}{intro_i(as, (b_k)_k) : P(ps_i[as])},$$

$$\frac{as = as' :: Gs_i \qquad (b_k = b'_k : \Pi zs :: Hs_{ik}[as].P(qs_{ik}[as, zs]))_k}{intro_i(as, (b_k)_k) = intro_i(as', (b'_k)_k) : P(ps_i[as])},$$

where

- Gs_i is a telescope relative to T;
- $Hs_{ik}[xs]$ is a telescope relative to T in the context $xs :: Gs_i$ for each k;
- $qs_{ik}[xs, zs] :: Is$ relative to T in the context $xs :: Gs_i, zs :: Hs_{ik}[xs]$ for each k;
- $ps_i[xs] :: Is$ relative to T in the context $xs :: Gs_i$.

5.3 Inductive families in set theory

Let I and U be sets and let Φ be a rule set on $I \times U$. Then Φ inductively defines a family $\mathcal{IF}(\Phi)$ of sets in U over I by

$$\mathcal{IF}(\Phi)(i) = \{u \in U | \langle i, u\rangle \in \mathcal{I}(\Phi)\}$$

for each $i \in I$.

5.4 Interpretation of expressions

Interpretation of set expressions:

$$[\![P(as)]\!]\rho \;=\; \mathcal{IF}(\Phi_P)([\![as]\!]\rho),$$

where

$$\begin{aligned}\Phi_P \;=\; \bigcup_i \{ & \frac{\bigcup_k \{\langle [\![qs_{ik}[xs,zs]]\!]^{us\,ws}_{xs\,zs}, v_k(ws)\rangle | ws \in [\![Hs_{ik}[xs]]\!]^{us}_{xs}\}}{\langle [\![ps_i[xs]]\!]^{us}_{xs}, \langle |intro_i|, us, (v_k)_k\rangle\rangle} | \\ & us \in [\![Gs_i]\!], \\ & (v_k \in [\![Hs_{ik}[xs]]\!]^{us}_{xs} \to U)_k\},\end{aligned}$$

where U is chosen so that Φ_P is a rule set on $[\![Is]\!] \times U$. Such a $U = V_\alpha$ is found in a similar way to the case for inductive sets above.

Interpretation of element expressions:

$$[\![intro_i(as, (b_k)_k)]\!]\rho \;=\; \langle |intro_i|, [\![as]\!]\rho, ([\![b_k]\!]\rho)_k\rangle.$$

5.5 Soundness of the inference rules

Formation rule. This is sound since $\mathcal{IF}(\Phi_P)$ is a family of sets over $[\![Is]\!]$.

The introduction rule is sound because assume that

$$us \in [\![Gs_i]\!]\rho$$

and

$$v_k \in \prod_{ws \in [\![Hs_{ik}[xs]]\!]\rho^{us}_{xs}} [\![P(qs_{ik}[xs, zs])]\!]\rho^{us\,ws}_{xs\,zs},$$

whenever $\rho \in [\![?]\!]$, for all k. Then it follows that for each $ws \in [\![Hs_{ik}[xs]]\!]\rho^{us}_{xs}$ we have

$$\langle [\![qs_{ik}[xs, zs]]\!]\rho^{us\,ws}_{xs\,zs}, v_k(ws)\rangle \in \mathcal{I}(\Phi_P).$$

Hence, by Φ_P-closedness

$$\langle [\![ps_i[xs]]\!]\rho^{us}_{xs}, \langle |intro_i|, us, (v_k)_k\rangle\rangle \in \mathcal{I}(\Phi_P)$$

and thus

$$\langle |intro_i|, us, (v_k)_k\rangle \in [\![P(ps_i[xs])]\!].$$

6 Primitive recursive families of functions

We give a schema for functions which are defined by recursion on the way the elements of $P(as)$ are generated. This generalizes the schema in section 4. Note that we have a kind of simultaneous recursion: an element of $P(ps_i[as])$ is generated from the elements of $(P(qs_{ik}[as, zs]))_k$.

6.1 Syntax

$$a ::= f(as)$$

6.2 Inference rules

Elimination rule:

$$\frac{as :: Is}{f(as) : \Pi z : P(as).C[z]},$$

$$\frac{as = as' :: Is}{f(as) = f(as') : \Pi z : P(as).C[z]}.$$

The ith equality rule:

$$\frac{as :: Gs_i \qquad (b_k : \Pi zs :: Hs_{ik}[as].P(qs_{ik}[as, zs]))_k}{\begin{array}{ll} & f(ps_i[as])(intro_i(as, (b_k)_k)) \\ = & d_i(as, (b_k, (\lambda zs :: Hs_{ik}[as].f(qs_{ik}[as, zs])(b_k(zs))))_k) \\ : & C[ps_i[as], intro_i(as, (b_k)_k)] \end{array}},$$

where

- $C[xs, z]$ is a set in the context $xs :: Is, z : P(xs)$;
- $$\begin{array}{lll} d_i & : & \Pi xs :: Gs_i. \\ & & (\Pi y_k : (\Pi zs :: Hs_{ik}[xs].P(qs_{ik}[xs, zs])). \\ & & \Pi y'_k : (\Pi zs :: Hs_{ik}[xs].C[qs_{ik}[xs, zs], y_k(zs)]))_k. \\ & & C[ps_i[xs], intro_i(xs, (y_k)_k)]. \end{array}$$

6.3 Primitive recursive families of functions in set theory

The rule set Φ_P is still deterministic. As a consequence we could define functions on the pairs $\langle as, c\rangle$ in $\mathcal{I}(\Phi_P)$. But we want curried versions instead. Such functions can be defined as inductive families of set-theoretic functions.

6.4 Interpretation of expressions

Let

$$[\![f(as)]\!]\rho \;=\; \mathcal{IF}(\Psi_f)([\![as]\!]\rho),$$

where

$$\begin{aligned}\Psi_f \;=\; \bigcup_i \{ & \frac{\bigcup_k \{\langle [\![qs_{ik}[xs,zs]]\!]^{us\,ws}_{xs\,zs}, \langle v_k(ws), v'_k(ws)\rangle\rangle | ws \in [\![Hs_{ik}[xs]]\!]^{us}_{xs}\}}{\langle [\![ps_i[xs]]\!]^{us}_{xs}, \langle\langle |intro_i|, us, (v_k)_k\rangle, [\![d_i]\!](us, (v_k, v'_k)_k)\rangle\rangle} | \\ & us \in [\![Gs_i]\!], \\ & (v_k \in \prod_{ws \in [\![Hs_{ik}[xs]]\!]^{us}_{xs}} [\![P(qs_{ik}[xs,zs])]\!]^{us\,ws}_{xs\,zs}, \\ & \;v'_k \in \prod_{ws \in [\![Hs_{ik}[xs]]\!]^{us}_{xs}} [\![C[qs_{ik}[xs,zs], y_k(zs)]]\!]^{us\,ws\,v_k}_{xs\,zs\,y_k})_k\}.\end{aligned}$$

Ψ_f is a rule set on $\sum_{us \in [\![Is]\!]} \sum_{w \in [\![P(xs)]\!]^{us}_{xs}} [\![C[xs,z]]\!]^{us\,w}_{xs\,z}$.

6.5 Soundness of inference rules

Omitted.

7 Polymorphism

7.1 Generic set formers and parameters

So far we have presented a completely monomorphic version of type theory similar to the type theory presented in Martin-Löf [9]. (Martin-Löf also introduced a new constant P for each instance of $\Pi x : A_0.A_1[x]$ and a new constant f for each instance $\lambda x : A.a[x]$. Thereby bound variables were avoided altogether.) But it is important both for convenience and expressiveness to introduce parameters in the inductive definitions. (We distinguish *generic set formers*, the arguments of which are called parameters, and *inductive families of sets*, the arguments of which are called indices.) A parameter can be either a set (or family) or an element (or function). The standard set formers all have sets (or families) as parameters: these are the A, A_0, and A_1 in $\Sigma x : A_0.A_1$, $A_0 + A_1$, $Eq(A)$, $Wx : A_0.A_1$. A nice example of a set former which has an element as a parameter is the equality predicate (due to Christine Paulin) $Eq'(A,a)$ on a set A, where the set A and the element $a : A$ are parameters. The elimination rule (disregarding proof objects) for this predicate is the rule of substitution: if C is a predicate on A such that $C(a)$ is true, then $\forall z : A.(Eq'(A,a)(z) \supset C(z))$. This rule is a derived rule for $Eq(A)$, which has a more complex elimination rule expressing that $Eq(A)$ is defined as the least reflexive relation on A. Another example of a set former which takes an element as a parameter is Peterson's and Synek's trees (generalized well-orderings) [11].

The set-theoretic interpretation extends directly to the case with parameters.

7.2 Typical ambiguity

The set-theoretic interpretation given above is polymorphic (introduces typical ambiguity) in the constructors but not in the recursive functions. This is because the denotation of $intro_i(as, (b_k)_k)$

does not depend on the denotations of Gs_i, and $Hs_{ik}[xs]$, and (in the case of families) Is, whereas the denotation of f depends on the denotations of Gs_i, $Hs_{ik}[xs]$, and d_i, and (in the case of families) Is, $ps_i[xs]$, and $qs_{ik}[xs, zs]$.

7.3 Internalization

I have presented an open theory, that is, a theory which can be extended whenever there is a need for it. But the schema precisely determines what an admissible extension of a given theory T is, and hence the schema defines a collection of theories obtainable by such extensions.

Is it possible to turn this external schema into an internal construction? Consider first the case of *inductive sets*. Each set P is determined by a list (indexed by i) of pairs of telescopes Gs_i and lists (indexed by k) of telescopes Hs_{ik}. It is tempting to write something like

$$P = \mathcal{W}((xs :: Gs_i, (Hs_{ik}[xs])_k)_i)$$

and

$$f = \mathcal{T}((xs :: Gs_i, (Hs_{ik}[xs])_k, d_i)_i),$$

because of the similarity between the schema for inductive sets and structural recursive functions on the one hand and the rules for the well-orderings on the other. We could then put

$$Wx : A_0.A_1[x] = \mathcal{W}(((x : A_0), ((A_1[x])))),$$

and observe that the well-orderings is the special case where all lists and telscopes have length 1. Other standard set formers could be defined by

$$\begin{aligned}
- &= \mathcal{W}(()), \\
\top &= \mathcal{W}((), ())), \\
N &= \mathcal{W}((), ()); (), ((), ()))), \\
\mathcal{O} &= \mathcal{W}((), ()); (), ((), ())); (), ((N)))), \\
A_0 + A_1 &= \mathcal{W}(((A_0), ()); ((A_1), ())), \\
A_0 \times A_1 &= \mathcal{W}(((A_0; A_1), ())), \\
\Sigma x : A_0.A_1[x] &= \mathcal{W}(((x : A_0; A_1[x]), ())).
\end{aligned}$$

However, there are no formal means in type theory for introducing $\mathcal{W}$ and $\mathcal{T}$, not even in the theory of logical types (Martin-Löf's logical framework). It seems that we would need to extend this framework with certain formal notions of telescope and non-dependent list.

Also note that we need the extra generality provided by the schema as compared with the well-orderings, since we consider *intensional* type theory. In *extensional* type theory on the other hand, we can use well-orderings for representing inductive sets, see Dybjer [6]. But even so, this is done by non-trivial coding and by assuming some basic set formers such as $-$, $\top$, $+$, and Σ, in addition to Π, which is needed for the schema too.

For *inductive families* we could similarly try to write

$$P = \mathcal{WF}(Is, (xs :: Gs_i, (zs :: Hs_{ik}[xs], qs_{ik}[xs, zs])_k, ps_i[xs])_i)$$

and

$$f = \mathcal{TF}(Is, (xs :: Gs_i, (zs :: Hs_{ik}[xs], qs_{ik}[xs, zs])_k, ps_i[xs], d_i)_i).$$

This does not resemble any standard set former, even though the first three arguments have a similar function to the first three arguments of Petersson's and Synek's trees [11].

Some set formers written in terms of $\mathcal{WF}$:

$$\begin{aligned} Eq(A) &= \mathcal{WF}((A),((x:A),(),(x;x))), \\ Eq'(A,a) &= \mathcal{WF}((A),((),(),(a))). \end{aligned}$$

8 Universes

We assume a countable sequence $U_0, U_1, U_2, \ldots$ and formulate rules for universes a la Russell [10].

Formation rules:

$$U_n \ set,$$

$$U_n = U_n.$$

Introduction rules:

$$U_m : U_n,$$

$$U_m = U_m : U_n$$

if $m < n$, and

$$\frac{? \vdash A_0 : U_n \qquad ?, x : A_0 \vdash A_1[x] : U_n}{? \vdash \Pi x : A_0.A_1[x] : U_n},$$

$$\frac{? \vdash A_0 = A_0' : U_n \qquad ?, x : A_0 \vdash A_1[x] = A_1'[x] : U_n}{? \vdash \Pi x : A_0.A_1[x] = \Pi x : A_0'.A_1'[x] : U_n}.$$

Elimination rules:

$$\frac{A : U_n}{A \ set},$$

$$\frac{A = A' : U_n}{A = A'}.$$

These rules extend the dependently typed λ-calculus with universes (getting the theory T_0^U) and are independent of the particular set formers introduced later. As in the case without universes we may extend T_0^U to obtain a sequence of theories $T_1^U, T_2^U, \ldots$. When extending $T = T_n^U$ to T_{n+1}^U by adding a new set former P or a new function constant f we may use the rules for universes to justify that the Gs_i, $Hs_{ik}[xs]$, etc. are sets.

Moreover, for each set former $P(ts)$ which may depends on certain parameters ts (see the previous section) we have universe introduction rules reflecting the formation rule of P. Assume that the definition of $P(ts)$ involves the telescopes $Gs_i[ts]$ and $Hs_{ik}[xs, ts]$ (see section 3 and 5), and that $Gs_i[ts] : U_n$ and $Hs_{ik}[xs, ts] : U_n$ $(xs :: Gs_i)$ whenever the set parameters in ts are in U_n. Then we get an introduction rule for U_n by modifying the conclusion $P(ts)$ set of the formation rule to $P(ts) : U_n$, by modifying each premise for a set parameter $A : set$ to $A : U_n$, and by leaving each premise for an element unchanged. (This situation is somewhat complex and we won't give a completely formal presentation. If we introduce the internal set former for inductive families $\mathcal{WF}$, then it would be easy to let the formation rule of $\mathcal{WF}$ be reflected as a universe introduction rule.)

A natural set-theoretic interpretation of the sequence of universes is as a sequence of set-theoretic universes. But this would not reflect the fact that a universe in type theory is a set,

and thus inductively defined by its introduction rules. In particular it would not interpret the elimination rule for that universe.

So instead we could make the interpretation dependent on the particular collection of set formers introduced, and let rule sets corresponding to the introduction rules for U_n inductively generate $[\![U_n]\!]$. The latter approach is similar to Salvesen's interpretation of the universe [12]. Note however that, provided we have introduced at least one infinite set in U_0, the requirement that $[\![U_0]\!]$ is a set implies that there exists a strongly inaccessible cardinal $\sup_{u \in [\![U_0]\!]} \mathrm{card}\, u$.

References

[1] P. Aczel. An introduction to inductive definitions. In J. Barwise, editor, *Handbook of Mathematical Logic*, chapter C.7, pages 739–782. North Holland, 1977.

[2] P. Aczel. The type theoretic interpretation of constructive set theory: inductive definitions. In *Logic, Methodology and Philosophy of Science VII*, pages 17–49. Elsevier Science Publishers B.V., 1986.

[3] R. Backhouse. On the meaning and construction of the rules in Martin-Löf's theory of types. Technical Report CS 8606, University of Groningen, Department of Mathematics and Computing Science, 1986. Presented at the Workshop on General Logic, Edinburgh, February 1987.

[4] R. Backhouse, P. Chisholm, G. Malcolm, and E. Saaman. Do-it-yourself type theory (part 1). *Formal Aspects of Computing*, 1:19–84, 1989.

[5] T. Coquand and C. Paulin. Inductively defined types. In *Proceedings of the Workshop on Programming Logic, Båstad*, May 1989.

[6] P. Dybjer. Inductively defined sets in Martin-Löf's type theory. In *Proceedings of the Workshop on General Logic, Edinburgh*, February 1987.

[7] P. Dybjer. An inversion principle for Martin-Löf's type theory. In *Proceedings of the Workshop on Programming Logic, Båstad*, May 1989.

[8] P. Martin-Löf. Hauptsatz for the intuitionistic theory of iterated inductive definitions. In J. E. Fenstad, editor, *Proceedings of the Second Scandinavian Logic Symposium*, pages 179–216. North-Holland, 1971.

[9] P. Martin-Löf. An intuitionistic theory of types: Predicative part. In *Logic Colloquium '73*, pages 73–118. North-Holland, 1975.

[10] P. Martin-Löf. *Intuitionistic Type Theory*. Bibliopolis, 1984.

[11] K. Petersson and D. Synek. A set constructor for inductive sets in Martin-Löf's type theory. In *Category Theory and Computer Science*, pages 128–140. Springer-Verlag, LNCS 389, 1989.

[12] A. B. Salvesen. Typeteori - en studie. Technical report, Department of Computer Science, University of Oslo, 1984. Cand.Scient-thesis.

[13] P. Schroeder-Heister. A natural extension of natural deduction. *Journal of Symbolic Logic*, 49(4), December 1984.

[14] P. Schroeder-Heister. Judgements of higher levels and standardized rules for logical constants in Martin-Löf's theory of logic. Unpublished paper, June 1985.

[15] T. Streicher. *Correctness and Completeness of a Categorical Semantics of the Calculus of Constructions*. PhD thesis, Fakultät für Mathematik und Informatik, Universität Passau, 1988.

[16] A. S. Troelstra. On the syntax of Martin-Löf's type theories. *Theoretical Computer Science*, 51:1–26, 1987.