Investigations Into Intensional Type Theory

Thomas Streicher

November 1993

Habilitationsschrift

vorgelegt der Fakultät für Mathematik
der Ludwig-Maximilians-Universität München

von Thomas Streicher

München, im November 1993

Table of Contents

Introduction

We first discuss the relevance of theories of dependent types for theoretical computer science. Then we explain the motivation for our work and finally give a suryey of the contents of our work.

The Relevance of Constructive Theories of Dependent Types for Theoretical Computer Science

In theoretical computer science the issue of *correctness of programs* has become more and more important over the years. Early approaches to establishing correctness were *program logics* such as Hoare calculi for verifying imperative programs and the method of *algebraic specification* for specifying modules in functional languages.
Program logics as e.g. the Hoare calculus are oriented towards state-oriented programming languages and provide calculi for deriving correct 'Hoare triples', i.e. expressions of the form $\{P\}S\{Q\}$ with the intuitive meaning that whenever the initial state of a program satisfies proposition P then the execution of program S terminates and results in a state satisfying proposition Q. Although this method works quite well for 'pidgin' programs (i.e. using only *assignment*, *if_then_else* and *while* over a *single* data type) there are big problems when dealing with procedures, modules and general user defined data types. In general a main drawback is that that the world of programs and the world of specifications are conceptually quite different. Of course, this problem is typical for all state-oriented imperative programming languages.
This conceptual gap between programs and their specifications has been overcome by the appearance and development of functional programming languages. As in functional programming all objects are either functions or tuples (maybe in a nested way) the world of programs and data appears as part of the term language of the specification language and therefore specifications of programs or modules can be expressed simply by logical formulae. The method of algebraic specification is from its very beginning oriented towards the specification of modules in functional programming languages. The basic intuition behind is that any such *module* in a functional programming language can be considered as a (constructive) *many-sorted algebra* where the *types of the module* are conceived as the *carrier sets of the algebra* and the *functions provided by the module* are conceived as the *operations of the algebra*. Thus according to the underlying philosophy of the algebraic specification method a specification of a module is given - at least in principle - by the theory of its corresponding many-sorted algebra.

As long as only first order operations are considered and the type system of the programming language is not too refined (a situation that can be found very often in practice !) this method works very well and in a pleasingly simple way. Traditionally the logical systems for representing the theory characterizing a module were and still are restricted to equational or Horn logic. More courageous people also considered full first order logic together with rules for structural induction and in principle one also can add higher order logic which allows to express induction schema by second order quantification. Nevertheless the intentional restriction to weak logics as equational or Horn logic has the advantage that it facilitates the model theory tremendously. Typically with any such theory one can associate the initial or the terminal model of the theory as the intended module described by the specification.
Nevertheless there is one main drawback of the algebraic specification method : one can build new functions from the given basic operations only in a very restricted way by explicit definition. But definition of new function(al program)s by structural or general recursion is not possible. But, of course, one can perform such an extension by adding a λ-calculus with a least fixpoint operator. This actually has been done within the LCF (*L*ogic of *C*omputable *F*unctionals) system. Variants of LCF have been implemented in Edinburgh and Cambridge and these systems have been widely used for the structured and verified development of quite complicated algorithms as e.g. unfication procedures.
But still, even this very powerful approach has two drawbacks. The first is that modules do not appear as first order citizens. In other words one can always deal only with the description and construction of one single module. The reason for this defect is that e.g. the type of all algebraic structures of a certain signature cannot be formulated without employing the concept of *dependent types*. The best way to explain the notion of dependent type is maybe by example (where in the following Σ denotes disjoint union of a family): $(\Sigma\ G : \mathrm{Set})\ G \to G \to G$ is the type of all structures with one type G and one (curried) binary operation ; $(\Sigma\ G : \mathrm{Set})\ G$ is the type of all pointed sets ; $(\Sigma\ G : \mathrm{Set})\ (\Sigma\ m : G \to G \to G)\ G$ is the type of all algebraic structures with one carrier set, one (curried) binary operation and one distinguished object. Now, of course, on the latter structure type one can define a predicate telling which such structures are a monoid, i.e. satisfy the laws of associativity and neutrality.
After this ‘explanation by example’ we will now explain abstractly what ‘dependent types’ mean in terms of analogy to common set-theoretic notions.
The notion of dependent type corresponds to the familiar notion of ‘family of sets indexed by a set’ as well-known from set-theoretic mathematics. Examples of ‘families of sets’ are abundant both in mathematics and computer science. Therefore - although all these concepts can in principle be expressed in (untyped) set-theory - at least for computer science it is better to have a typed language which generally supports structured presentations and modularity.
The main aspect of dependently typed languages is that *type expressions may depend on object expressions*. In dependently typed languages one can easily express *structure types* whose *objects are modules* and *predicates* and *operations* on them. Thus for languages supporting the construction and verification of large modular systems the concept of dependent type is inevitable.

But there is a further more foundational aspect of dependent types which historically was the actual motivation for logicians like Per Martin-Löf and J.Y.Girard to introduce their type theories at the beginning of the 1970s.
This foundational aspects of constructive type theories has become known as the *paradigm* of *propositions-as-types*. The importance of the the paradigm of propositions-as-types originates from the importance of *proofs* when applying formal methods. The key idea is to allow *proofs as objects* which can be *denoted by terms of the language*. Consequently a *proposition* is considered as the *type of its proofs* ! Such types are called propositional types. This gives rise to a *constructive interpretation of logic* where a proof of the proposition 'A *implies* B' is simply a *function* mapping proofs of A to proofs of B , *universal quantification* corresponds to products of families of propositional types and existential quantification to disjoint unions or sums of families of propositional types. In the end it appears that logic is included into a functional programming language and the task of proving appears as a subdiscpline of functional programming.
As the logic of type theory is constructive proof objects carry algorithmic information and therefore functional programs can be extracted from proofs.. The typical situation is the following. Let A and B be sets of inputs and outputs, respectively, $P \in A \rightarrow B \rightarrow \text{Set}$ be a predicate specifying the relation between inputs and outputs and suppose we have a proof object $p \in (\Pi x{:}A)(\Sigma y{:}B)\ P\ x\ y$. Then $f_{prog} \equiv (\lambda x{:}A)\ \pi_1(p) \in A \rightarrow B$ is the program extracted from p and $f_{corr} \equiv (\lambda x{:}A)\ \pi_2(p) \in (\Pi x{:}A)\ P\ x\ (f\ x)$ is the correctness proof for f_{prog} . Thus there is a uniform method of extracting programs out of proofs together with proofs for their correctness. This way all conceivable *algorithmic functions* can be obtained but in general the *algorithms* obtained in this way are not very efficient as they have to manipulate a lot of computationally irrelevant logical meta-information which is relevant for the correctness proof but should not show up in the algorithm itself. Therefore it is often more efficient to construct the programs in advance and verify them afterwards. From this point of view a specification of a functional program mapping A to B and satisfying predicate P is given by the *type* $(\Sigma f{:}A \rightarrow B)\ P(f)$. Thus an object of type $(\Sigma f{:}A \rightarrow B)\ P(f)$ is a pair (f,p) where $f : A \rightarrow B$ and p is an object in P(f) providing a proof (object) for the correctness of the functional program f .
The original aim of type discipline was to provide a *partial* correctness information by specifying e.g. the types of inputs and outputs. But the type systems which are implemented in functional programming languages as e.g. SML are to weak to express more realistic and therefore more restrictive correctness properties as e.g. " f computes the gcd of two natural numbers" by a type. The restricted expressiveness of these type systems lies in the fact that they do not allow dependent types which are required to include constructive logic into the programming language. Therefore the relevance of constructive theories of dependent types is that *program correctness reduces to type correctness* and that *there are efficient algorithms for type checking*. But, of course, as miracles are out of scope the correctness proof has to be supplied together with the program.

Motivation

The topic of this work is the investigation of *Intensional Type Theory* or *Intensional Constructive Set Theory* (ICST) introduced by Per Martin-Löf as a *foundation of constructive mathematics* and a *formal basis for developing (always terminating) functional programs.* A very precise introduction to ICST can be found in the book by Nordström, Petersson and Smith [NPS]. That the formalism of ICST is most suitable for these purposes has been demonstrated in [NPS] .
But the practically most relevant aspect of ICST beyond its foundational issue is that there is a couple of *computerized support systems* for *constructing and executing proofs and programs.* Rather well known examples are the LEGO system (cf. [LP]) developed by R. Pollack in Edinburgh, the COQ system (cf [DFHHPW]) developed by a group around G. Huet at INRIA and the ALF system (cf [Mag]) developed by the Gotenburg type theory group.
All these systems are based on *intensional* type theory as otherwise *type checking would become undecidable* ! The reason is that in order to have type-checking decidable one needs to have a *confluent* and *terminating rewrite system* representing the operational semantics of the underlying term language of the type theory under consideration.
As we will show in Chapter 1 it is the

Reflection Rule $$\frac{\Gamma \vdash p \in \mathrm{Id}\,A\,t\,s}{\Gamma \vdash t = s \in A}$$

- distinguishing *extensional* from *intensional constructive set theory* - which leads to the non-confluence of the underlying rewrite system. Therefore the Reflection Rule has to be abandoned and one has to stick to ICST when one wants to have a nice support system.
One might think that what one really would like to have is extensional type theory (as described e.g. in [ML3]). And actually extensional type theory has been taken as the basis for the NuPrL system (cf. [Const]) implemented by the group of R. Constable at Cornell University. The NuPrL system is the oldest of the various support systems for type theory. But instead of dealing with *terms* relative to a context of declarations and definitions it has to deal with *derivations* of *judgements* in the *formal system of type theory.* Of course, it does not deal with derivations explicitely but constructs them using quite refined tactics. So in the implementation the main advantage of type theory is lost : the principle of *propositions-as-types* which allows to represent proofs by *terms of the object language* instead of representing them by *derivation trees on the meta-level.*
But as mentioned above the more recent and more comfortable support systems as LEGO, COQ, ALF etc. have to pay the price of supporting only a weaker theory, namely ICST with its characteristic distinction between *judgemental* and *propositional* equality.
Although in ICST one can do most of the things one wants to do one has to be careful in for-

mulating propositions because the extensional way of thinking - which most mathematicians and computer scientists are used to - sometimes allows to make formulations of propositions and their proofs much shorter. And certain theorems simply do not hold due to the lack of extensionality. A typical example is that from $t \in B\ a$ and $p \in Id\ A\ a\ b$ one is not allowed to conclude that $t \in B\ b$ where A is a type and B is a family of types indexed over A. (But, of course, one is allowed to infer $t \in B\ b$ from $t \in B\ a$ and $a = b \in A$!)
It is evident from the practical experience in using and teaching type theory that ICST is full of - sometimes unpleasant - surprises for the naive user. The reason is that the naive user is inclined to think in an extensional way because nowadays everybody is introduced to elementary mathematics in the language of naive set theory.
Accordingly, a lot of people have observed a couple of propositions which are provable in extensional type theory but have resisted any attempt to derive them in intensional type theory.

It is exactly these *independence problems* which will be studied in this thesis.

There are to kinds of answers which one can give to these questions.
The first kind of answers is of *syntactical character* and will be dealt with in Chapter 1. There we define extensions of current formulations of ICST, show that they are computationally meaningful and justified by their equivalence to propositions which intuitively should be valid.
The second kind of answers is of *semantical character* and will be dealt with in Chapters 2 and 3 . There we will show for various propositions that they are not only so difficult to derive that nobody has manged up to now but that these propositions are really underivable. When trying to show underivability one quickly finds out that *syntactic proofs of underivability* are considerably hard already for simple problems and as far as the author knows no one hardly ever has succeeded in giving such syntactic proofs for non-derivability. And, actually, the common practice in mathematics for obtaining *underivability results* is to *construct counter-models*.
That is the point where *mathematical models of type theory become relevant*. Besides providing the logical hygeny called consistency their main use is to provide *inspiration for extensions of current formulations of a formal system* and to *refute propositions which suspected to be underivable*.
It is this latter aspect of models we concentrate on. We have already previously given semantical independence proofs for extensional type theory (cf. [Str2] , [Str3]).
In this thesis we will give semantical proofs of underivability for most of those propositions which are derivable in extensional type theory but have resisted any attempt to derive them formally in ICST.
We furthermore suggest and give evidence by the examples we study that semantical methods are helpful for checking the validity of suggested extensions of ICST.

Survey of Contents

Chapter 1

Whereas the subsequent chapters are mainly concerned with semantical questions the first chapter provides syntactical motivation for these latter studies. Our interest in phenomena of intensionality has been initiated by trying to prove that all objects of an identity type are propositionally equal. This is quite easy to achieve in extensional CST as - at least in some formulations, e.g. [ML2],[ML3] - it is even put as an axiom that all object of type Id A a a are even judgementally equal to the canonical object r A a and in other more recent formulations of extensional type theory as e.g. in [NPS] it is easily obtained by the Reflection Rule.
But for the intensional theory - as described e.g. in [NPS] - it is up to now an open problem whether it can be proved that all objects of an identity type are propositionally equal. But a conceptual analysis of the problem shows that the problem can be solved by an obvious extension of the traditional formulation of intensional CST as described in [NPS]. One simply has to introduce an new additional eliminator for identity types.
Whereas the traditional eliminator

```
J ∈ {A : Set} {C : {x : A} {y : A} {z : Id A x y} Set}
    {d : {x : A} C x x (r A x)}
    {a : A} {b : A} {c : Id A a b}
      C a b c
```

is the eliminator for the inductive family of sets corresponding to the *binary equality* predicate (on set A) our new additional eliminator

```
K ∈ {A : Set} {C : {x : A} {z : Id A x x} Set}
    {d : {x : A} C x (r A x)}
    {a : A} {c : Id A a a}
      C a c
```

corresponds to the eliminator for the inductively defined family of sets corresponding to the *unary* predicate obtained by *diagonalization of the binary equality predicate above.*
Chapter 1 of this work - which exclusively deals with syntactic questions - is mainly devoted to the study of this extension of intensional type theory.
After some preliminary observations about different formulations of extensional CST (especially we show that the Reflection Rule and the η-rule for J are equivalent) we discuss the problem of proving that all objects of an identity type are propositionally equal and then analyze the reasons for the apparent underivability of this proposition. We show that this problem can be solved quite easily after adding our new additional eliminator K . We explain why it is natural to add the new eliminator K and show that it is equivalent to the derivability of even more obvious propositions about equality of pairs. We finally relate our new elimina-

tor K to other new eliminators for identity types which have evolved in the type theory community and show that these suggestions can be subsumed under our extension by K .
Finally, referring to a recent observation of Th. Altenkirch, we show that in any "reasonable" model of intensional type theory the propositional equality of all objects of an identity type can be proven. Here "reasonable" means that terms of the same type are equal iff their underlying algorithms obtained by stripping off computationally irrelevant type information are equal.
We have chosen this specific extension of the current formulation of ICST for the following reasons.
Firstly it has been our personal motivation (initiated by a question of Th. Coquand) to enter the field of intensionality and - more important - the subtleties of intensional type theory are concetrated into the intriguing interplay between judgemental and propositional equality.
Secondly this extension is necessary for fullfilling the central claim of the underlying philosophy of intensional CST, namely that it should be derivable in the theory that any object a of an inductive type is propositionally equal to a canonical object c which is obtained by applying a uniquely determined constructor to the components of the object a . This is a generalization of *surjective pairing* or the *η-rule for functional types* to *arbitrary inductive types*.
The eliminator K can be interpreted in the models discussed in chapters 2 and 3 as easily as the eliminator J . Therefore we think this example strongly supports our claim that in order to check whether a suggested extension of intensional CST is reasonable a good test is trying to interpret it in the models we study in the subsequent chapters.

Chapter 2
This is the first chapter presenting a model which is non-extensional. It is based on previous work by P. Aczel [Acz]. Whereas he gives a model where one only can interpret Set , i.e. small types, we extend his model in a way that the ambient logical framework can be interpreted. The model is most concisely described as the *sconing* of *Freyd cover* of some *appropriate category of domains*. Expressed in elementary terms the underlying category can be described as follows. The objects are triples $\underline{X} = (X , D , r)$ where. X is an arbitrary set , D is an arbitrary domain and r is an arbitrary mapping associating with *any object* $x \in X$ *its realizer* $r(d) \in D$. A morphism from $\underline{X}_1 = (X_1 , D_1 , r_1)$ to $\underline{X}_2 = (X_2 , D_2 , r_2)$ is given by a set-theoretic function $f : X_1 \rightarrow X_2$ *together* with a domain morphism (e.g. a Scott continuous function or a stable function) $a : D_1 \rightarrow D_2$ such that $r_2(f(x)) = a(r_1(x))$ for all $x \in X_1$.
Intuitively that means that the domain morphism a maps the realizer for an argument to the realizer for the result. We say that a *realizes* the function and f is the *underlying set-theoretic function* of the type-theoretic function. Peter Aczel's original model can be recovered from our more general setting by restriction to those $\underline{X} = (X , D , r)$ where X is a subset of D and r is the inclusion map of X into D and D is some arbitrary (domain) model of (an extension of) untyped λ-calculus fixed in advance. But for interpreting e.g. the big type Set - as needed for interpreting the ambient logical framework - then one has to exploit the full generality of our setting. The type Set will be interpreted as the triple ($\mathcal{P}(D)$, 1 , r :

$\mathcal{P}(D) \to 1$) where D is some domain model of (an extension) of untyped λ-calculus. Similarly the interpretation of a type of families of sets will also have as underlying domain a trivial one containing exactly one object. This trivial realizability structure reflects the intuition that elements of such big 'meta'-types do not carry any computaionally relevant information.
In Chapter 2 we will show that this structure provides a model for *predicative* Intensional Constructive Set Theory and satisfies the following two criteria of intensionality

(1) $A : \mathrm{Set}, x : A, y : A, z : \mathrm{Id}\ A\ x\ y \vdash x = y \in A$

is *not* valid

(2) $\vdash p \in \mathrm{Id}\ A\ t\ s$ implies $\vdash t = s \in A$

Furthermore we show that it is not valid in these models that functions which are pointwise equal are already equal as objects of their function type, i.e. the *extensionality principle for functions fails*. Furthermore we show that in general functions defined on one of the usual inductive types may be different even if they give the same results when applied to objects in constructor form. Therefore the η-rules for inductive types - which are typical for extensional type theory - fail in our models as they state the uniqueness of elimination, i.e. that functions on an inductive type are determined uniquely by their behaviour on objects in constructor form.
One drawback of the model studied in Chapter 2 is that it can be extended to a model of *impredicative* ICST only in a restricted sense.
A more crucial drawback of this model is that it validates the following sequent

(3) $A : \mathrm{Set}, B : A \to \mathrm{Set}, x : A, y : A, z : \mathrm{Id}\ A\ x\ y \vdash B\ x = B\ y$

which definitely should not hold for a fully intensional model. But sequent (3) has been assumed in the type theory employed by J. Smith in his Thesis [Smi]. This is not a big surprise as there he gave an axiomatization of type theory which is complete w.r.t. to a (syntactic) interpretation in a so-called theory of constructions, i.e. a first order constructive theory of λ-calculus extended with pairing, natural numbers etc.

Chapter 3
To overcome these drawbacks in Chapter 3 we introduce a fully intensional model of ICST satisfying the criteria (1) and (2) and furthermore refutes the sequent (3) and allows to interpret impredicative ICST where the type Set is closed under arbitrary dependent products !
A particularly pleasing aspect of this model is its simplicity. It is obtained only by a slight modification of the well-known *realizability model for extensional type theory* based on **ω-Set** which was originally introduced by E. Moggi and studied to great detail by a lot of authors (cf. e.g. [Str1]). The suitability and flexibility of **ω-Set** as a model for extensional type

theory has been exploited especially in order to obtain independence results (cf. [Str2],[Str3]). The main advantage of the realizability model is that one can compute in it almost like in **Set** - one only has to provide a realizer for any function defined in terms of naive set theory and as long as the definition is constructive enough one obtains a required realizer by appealing to Church's Thesis (which - with some effort - can be made into a precise argument !).

Now, instead of ω-Set - from now on called r-Set (acronym for realizability sets) - we employ the category mr-Set (acronym for modified realizability sets).
This alternative choice is not of a purely technical nature but has its origin in the following conceptual analysis.
What we want to refute in a non-extensional model of CST is that functions are already equal if they give the same result when applied to 'canonical' objects as arguments. On the syntactical level one can have the phenomenon that two function terms give the same result up to convertibility when applied to arbitrary *closed* terms of the correct type but are not equal, i.e. not convertible, when applied to an open term of the appropriate type as e.g. a variable of that type. When we want to mimick this syntactic phenomenon on the level of semantics we come to the conclusion that variables do not only range over 'canonical', i.e. *actual* objects, but also over so-called *potential* objects which are not really denotable by terms but which could be made denotable by some future extension of the syntax. Now if one accepts these *potential* objects then it may quite well be the case that two functions show the same behaviour on all actual objects but give different results when applied to non-actual potential objects.
This distinction between potential and actual objects arises from the ancient Greek idea that in order to prove a universal statement it is sufficient to prove the statement for a *most general object* of the appropriate type. The point is that it is impossible to give semantical status to the informal concept of a 'most general object'. Instead we allow an arbitrary collection of non-actual, only potential objects which *together* simulate the idea of a 'most general object' by modifying the above mentioned ancient Greek idea in the following way : *to prove a universal statement is to prove the statement for all potential objects and not only the actual ones* !

Based on this consideration we accordingly define the category mr-Set as follows : an mr-set is an r-set together with a distinguished subset of actual objects and morphisms between mr-sets are simply morphisms between the underlying r-sets which in addition preserve actual objects. This category serves as a model for the ambient logical framework which has to be extensional. The crucial point for full intensionality is how Set is interpreted. A *set* (in the sense of ICST) is an mr-set where *potential objects are uniquely determined by their realizers* and *there is always a (potential) object realized by 0 serving as a default or error element* allowing to refute universal propositions even if they are valid for all instances by canonical objects.
In Chapter 3 we demonstrate that this model is really fully intensional and that the extensionality principle fails for functions. Furthermore we show that the inductive type N of natural numbers is not isomorphic to a certain W-type which in the extensional theory is isomorphic

to N . We further show that that in our model the empty type N_0 and the singleton type N_1 do not satisfy several propositions which intuitively seem to be valid and can be proven in the extensional theory.
Due to the fact that the model based on mr-**Set** allows to interpret impredicative universal quantification one has Leibniz equality types in parallel with Martin-Löf's identity types. We investigate the relation between these two concepts of propositional equality. First we show that in the Extended Calculus of Constructions they are not even equivalent if Martin-Löf's identity types do not live on the level of the type Prop but on the level of Type(0) . If both notions of propositional equality live on the level of the impredicative type Set then we show that - also they are equivalent - they are not isomorphic. The reason for this phenomenon is that for Martin-Löf identity types realizers for actual objects are recursively separable from realizers for non-actual objects whereas this is not the case for Leibniz equality types.
Finally we show that sets in the sense of ICST can also be interpreted as some kind of effective domains (complete extensional pers with bottom, cf. [FMRS]) with a distinguished subcollection of actual objects. For this interpretation of Set it is shown that Σ cannot be interpreted.

Chapter 1

Syntactic Aspects of Intensional Constructive Set Theory

In this chapter we will discuss syntactic aspects of Intensional Constructive Set Theory.
In the first section we show that the distinguishing feature of intensional CST is that the notions of judgemental and propositional equality are not equivalent anymore as in the extensional theory. We show that the reflection rule of extensional CST which makes both notions of equality equivalent can be expressed in a purely equational way by an η-rule for the eliminator J for Martin-Löf's Identity Sets.
In the second section we discuss the apparent incompleteness of the current formulation of ICST w.r.t. the derivability of the proposition expressing that all objects p of set Id A a a are are propositionally equal to the canonical object r A a . We introduce a new eliminator K for identity sets and some equivalent formulation and show that K is equivalent to several intuitively valid propositions.
In section 3 we show that another eliminator for identity sets suggested by Ch. Paulin-Mohring can be expressed by our K and cite a proof of the fact that under under the assumption of impredicativity for Set Paulin-Mohring's eliminator can be defined already in terms of Martin-Löf's J .
In section 5 we generalize our new eliminator K by introducing the concept of identity sets for set contexts where with an identity set for a context of length n there are associated n+1 eliminators. In the case of n=1 the two eliminators correspond to Martin-Löf's eliminator J and our eliminator K .
In section 6 we show that under the assumption of a so-called uniformity principle the eliminator K can be expressed in terms of J . The uniformity principle which is valid in all known models which are not term models claims that any two terms of the same type whose underlying algorithms are convertible are already judgementally equal.
Finally in section 7 we discuss our extensions of ICST and relate it to the perspective of a more liberal conception of ICST introduced by Th. Coquand where intensional type theory is conceived as an open system allowing definition schemes extending structural recursion.

1.1 Intensional versus Extensional Constructive Set Theory

Over the years Martin-Löf's Constructive Set Theory has undergone a lot of changes. The main changes were motivated by the following questions.

(1) Should one distinguish between intensional and extensional equality ?

(2) How can the textual macro structure of mathematical theories itself be expressed in a type theoretic language ?

The answer to the second question has led to the development of so called *logical frameworks* which have been implemented quite successfully and allow to "declare" ones favorite type theory in very high level languages, namely the logical frameworks, which - in the essence - are the core of deBruijn's AUTOMATH languages, i.e. a dependently typed λ-calculus with product types, the so called Π-types .

To the first question - which will be our main concern here - the final answer has not yet been given and probably extensional and intensional constructive set theory have merits of their own.

Surely, intensional constructive set theory is not as well understood as the extensional one - at least from the semantical point of view. But it is of great practical relevance as the shift to intensional constructive set theory facilitates the construction of computerized support systems for constructive set theories.

Now, of course, one might think of intensional constructive set theory as the *best approximation* to extensional constructive set theory which still satisfies normalisation, decidability of type checking etc. But if one wants to understand the *defects* of intensional constructive set theory (w.r.t. expressivity compared with extensional constructive set theory) it seems to be inevitable to have nice models at hand in order to derive independence results.

But before one can construct models one first has to understand the conceptual differences between *propositional* and *judgemental equality.* Actually, an understanding of these differences is necessary even for the most simple minded use of Martin–Löf's *identity sets.*

Quite generally one can say that in constructive set theory the notion of *identity* or *equality* appears in two different forms. On the one hand as a judgement of the form $t = s \in A$ stating that " t *and* s *are both objects of set* A *and definitionally equal as such* " and on the other hand as the proposition(al set) Id A t s whose objects are the proof objects or realizers for the proposition that " t and s are equal objects of set A " . If we know that Id A t s is a set we also know that $t \in A$ and $s \in A$ which, of course, does not at all imply that there is an object contained in the set Id A t s .

In all versions of constructive set theory one has *reflexivity* of propositional equality, i.e. one can prove from the judgement $t = s \in A$ that for some object p we have $p \in$ Id A t s . By meta-mathematical reasoning it follows that if $p \in$ Id A t s is derivable w.r.t. the *empty context* then $t = s \in A$ is derivable w.r.t. the empty context as well. Thus - when reasoning absolutely, i.e. not using any non-analytical material assumptions - then the two concepts of identity do coincide. This latter meta-mathematical property has been discussed in Z. Luo's Thesis [L1] and baptized there as "equality reflection lemma".

Alas, the situation is quite different when reasoning w.r.t. non-empty contexts where the existence of some mathematical objects or the validity of some mathematical facts is assumed which are not guaranteed by purely logical reasons. A simple, but typical example is the following. Consider the context $x : N$ (where N is the type of natural numbers) and relative to this context the terms $t \equiv [x : N]\, x$ and $s \equiv [x : N]\, R\, 0\, ([x : N][y : N]\, \mathrm{succ}\, y)\, x$ which represent *different ways of computing* the identity function on natural numbers. Then there is no way to derive the sequent $x : N \vdash t\, x = s\, x \in N$ as there is no way to *convert* or *rewrite* the term $R\, 0\, ([x : N][y : N]\, \mathrm{succ}\, y)\, x$ to the term x. This phenomenon is intended as t and s are intensionally different in the sense that they correspond to different algorithms computing the identity function on natural numbers. Nevertheless - using induction - we can construct a term p such that $x : N \vdash p \in \mathrm{Id}\, N\, (t\, x)\, (s\, x)$. The latter judgement expresses that whenever one substitutes for x a *canonical object* c of type N then $t\, c$ and $s\, c$ both reduce to the same canonical element of type N. That means that *for any choice of concrete values for the variables in the context* - respecting the type requirements of the context - the instances of $t\, x$ and $s\, x$ are equal even in the sense of definitional identity.

Now should one consider this as a defect of the theory or as something intended ? Perhaps this is a matter of taste, but anyway, there are two different ways of coping with that problem !
One is to accept the difference between these two notions of identity as done in the original version of constructive set theory, see [ML1], and in the current 'official' one, see [NPS], which results in the so called *intensional constructive set theory.* The other one is to (try to) remedy this 'defect' by adding further rules as e.g. in [ML2], [ML3] which results in the so called *extensional constructive set theory.*

The only rule which distinguishes extensional from intensional set theory is the so called

Reflection Rule

$$\frac{\Gamma \vdash p \in \mathrm{Id}\, A\, t\, s}{\Gamma \vdash t = s \in A}$$

which does not fit into the aesthetically appealing pattern of introduction and elimination rules. As we shall see immediately quite serious problems are caused by this rule !
Of course, there is no problem at all with consistency but adding the Reflection Rule has an enormous impact on the computational behaviour of the underlying functional language.
If one thinks that computation essentially is (applying) conversion (rules) then an elementary computation step might require to prove that considerably abstract objects as e.g. functions between natural numbers are equal. As we know such proofs may be arbitrarily complicated. Actually, it is exactly this class of propositions wherein arithmetic becomes incomplete (the famous Gödel sentence is of the form $(\forall n)\, \mathrm{proves}(n,\text{'G'}) = 0$ where proves is a binary primitive recursive function and 'G' is the Gödel number of the Gödel sentence).

In order to understand the special status of the Reflection Rule we first have to reconsider the general scheme of the introduction and elimination rules for non-circular inductive sets as e.g.

Π-, Σ- or Id-sets (for circular inductive sets see [Dyb]).

For each *non-circular inductive set* finitely many introduction rules describe how to construct canonical elements of this set.
In the logical framework approach this means that with any set A there is associated a finite number of *constructors*

$$\mathrm{intro}_i \in \{y_{i,1} : B_{i,1}\} \ldots \{y_{i,j_i} : B_{i,j_i}\}\ A \qquad (\, i \in [1\,,n]\,)$$

By definition a *canonical object* in A is given by a term of the form $\mathrm{intro}_i\ c_1 \ldots c_{j_i} \in A$ for some $i \in [1\,,n]$ and canonical objects $c_1 \in B_{i,1}\,, \ldots, c_{j_i} \in B_{i,j_i}[c_1\,, \ldots, c_{j_i-1} / y_{i,1}, \ldots, y_{i,j_i-1}]$.
Furthermore, for any family C of sets indexed over the set A and any mappings

$$d_i \in \{y_{i,1} : B_{i,1}\} \ldots \{y_{i,j_i} : B_{i,j_i}\}\ C\,(\mathrm{intro}_i\ y_{i,1} \ldots y_{i,j_i}) \qquad (\, i \in [1\,,n]\,)$$

there is a canonical mapping

$$\mathrm{Elim}\ d_1 \ldots d_n \in \{x : A\}\ C\,x$$

whose behaviour on objects in constructor form - and therefore on canonical objects - is as prescribed by the mappings d_i, i.e. for all $i \in [1\,,n]$ and $y_{i,1} \in B_{i,1}\,, \ldots, y_{i,j_i} \in B_{i,j_i}$

$$\mathrm{Elim}\ d_1 \ldots d_n\,(\mathrm{intro}_i\ y_{i,1} \ldots y_{i,j_i}) = d_i\ y_{i,1} \ldots y_{i,j_i} \in C\,(\mathrm{intro}_i\ y_{i,1} \ldots y_{i,j_i})$$

This situation is illustrated by the following diagram where proj stands for the canonical projection of the context $x : A\,, z : C\,x$ to the context $x : A$ and the map $\mathrm{Elim}\ d_1 \ldots d_n$ is understood to be a section of proj, i.e. $\mathrm{proj} \circ \mathrm{Elim}\ d_1 \ldots d_n = \mathrm{id}_{x:A}$, and the diagm is supoosed to hold for all $i \in [1\,,n]$.

$$\begin{tikzcd}
 & x : A\,, z : C\,x \arrow[d, shift left, "\mathrm{proj}"] \\
y_{i,1} : B_{i,1}\,, \ldots, y_{i,j_i} : B_{i,j_i} \arrow[ur, "d_i"] \arrow[r, "\mathrm{intro}_i"'] & x : A \arrow[u, shift left, "\mathrm{Elim}\ d_1 \ldots d_n"]
\end{tikzcd}$$

Now as the very assumption of constructive set theory is that *all objects in a set are equal to an object in constructor form* it is natural to assume that a function is uniquely determined by its behaviour on canonical elements as long as we insist on extensional equality, where two functions are equal iff they give the same value for all arguments.
This suggests the following *uniqueness of elimination* rule for objects of set A :

If $f, g \in \{x : A\}\, C\, x$

and for all $i \in [1, n]$ and $y_{i,1} \in B_{i,1}, \ldots, y_{i,j_i} \in B_{i,j_i}$

$$f\,(\text{intro}_i\; y_{i,1} \ldots y_{i,j_i}) = g\,(\text{intro}_i\; y_{i,1} \ldots y_{i,j_i}) \in C\,(\text{intro}_i\; y_{i,1} \ldots y_{i,j_i})$$

then $f = g \in \{x : A\}\, C\, x$

An alternative way to express uniqueness of elimination is to claim the following equivalent rule which strongly resembles the η-rule known from untyped λ-calculus :

For any $f \in \{x : A\}\, C\, x$

$$\text{Elim} \ldots ([y_{i,1} : B_{i,1}] \ldots [y_{i,j_i} : B_{i,j_i}]\; f\,(\text{intro}_i\; y_{i,1} \ldots y_{i,j_i})) \ldots = f \in \{x : A\}\, C\, x\ .$$

Not all of these rules expressing uniqueness of elimination for the various inductively defined sets of Martin-Löf's extensional Constructive Set Theory are part of the traditional formulation as can be found e.g. in [ML2] or [ML3]. Actually, it is sufficient to claim uniqueness of elimination for identity sets only as this is equivalent to the ordinary Reflection Rule as shown in the following Theorem 1.1 .

Theorem 1.1

If we add to intensional constructive set theory the following axiom expressing uniqueness of elimination for identity sets

$$\begin{array}{l} A : \text{Set}\ ,\ C : \{x : A\}\, \{y : A\}\, \{z : \text{Id}\, A\, x\, y\}\ \text{Set}\ , \\ e : \{x : A\}\, \{y : A\}\, \{z : \text{Id}\, A\, x\, y\}\, C\, x\, y\, z \\ \vdash \\ J\, A\, C\, ([x : A]\, e\, x\, x\, (r\, A\, x)) = e \in \{x : A\}\, \{y : A\}\, \{z : \text{Id}\, A\, x\, y\}\, C\, x\, y\, z \end{array}$$

then we can derive the Reflection Rule.

Proof : Assume uniqueness of elimination for J . Let $A \in \text{Set}$. Then we define a family $C := [x : A]\, [y : A]\, [z : \text{Id}\, A\, x\, y]\, A$ of type $\{x : A\}\, \{y : A\}\, \{z : \text{Id}\, A\, x\, y\}$ Set and two functions

$$\text{pr1} := [x : A]\, [y : A]\, [z : \text{Id}\, A\, x\, y]\ x \in \{x : A\}\, \{y : A\}\, \{z : \text{Id}\, A\, x\, y\}\, A$$

$$\text{pr2} := [x : A]\, [y : A]\, [z : \text{Id}\, A\, x\, y]\ y \in \{x : A\}\, \{y : A\}\, \{z : \text{Id}\, A\, x\, y\}\, A\ .$$

From the definition of pr1 and pr2 it is obvious that

$$[x : A]\ \mathrm{pr1}\ x\ x\ (r\ A\ x) = [x : A]\ x = [x : A]\ \mathrm{pr2}\ x\ x\ (r\ A\ x) \in (A)\ A$$

as we have

$$x : A \vdash \mathrm{pr1}\ x\ x\ (r\ A\ x) = ([x : A]\ [y : A]\ [z : \mathrm{Id}\ A\ x\ y]\ \ x)\ x\ x\ (r\ A\ x) = x$$

and

$$x : A \vdash \mathrm{pr2}\ x\ x\ (r\ A\ x) = ([x : A]\ [y : A]\ [z : \mathrm{Id}\ A\ x\ y]\ \ y)\ x\ x\ (r\ A\ x) = x \quad .$$

Thus we get by symmetry and transitivity that

$$J\ A\ C\ ([x : A]\ \mathrm{pr1}\ x\ x\ (r\ A\ x)) = J\ A\ C\ ([x : A]\ \mathrm{pr2}\ x\ x\ (r\ A\ x))$$

But from uniqueness of elimination for J we get that

$$\mathrm{pr1} = J\ A\ C\ ([x : A]\ \mathrm{pr1}\ x\ x\ (r\ A\ x)) \in \{x : A\}\ \{y : A\}\ \{z : \mathrm{Id}\ A\ x\ y\}\ A$$

$$\mathrm{pr2} = J\ A\ C\ ([x : A]\ \mathrm{pr2}\ x\ x\ (r\ A\ x)) \in \{x : A\}\ \{y : A\}\ \{z : \mathrm{Id}\ A\ x\ y\}\ A$$

and therefore again by transitivity and symmetry we get that

$$\mathrm{pr1} = \mathrm{pr2} \in \{x : A\}\ \{y : A\}\ \{z : \mathrm{Id}\ A\ x\ y\}\ A \ .$$

Now assume that $t \in A$ and $s \in A$ and $p \in \mathrm{Id}\ A\ t\ s$. Then we get immediately that

$$\mathrm{pr1}\ t\ s\ p = \mathrm{pr2}\ t\ s\ p \in A$$

but according to the definition of pr1 and pr2 we get that

$$\mathrm{pr1}\ t\ s\ p = t \in A \qquad \text{and} \qquad \mathrm{pr2}\ t\ s\ p = s \in A$$

and therefore by symmetry and transitivity finally that

$$t = s \in A \quad . \qquad \Diamond$$

The inverse direction claiming that the Reflection Rule implies the uniqueness of elimination is quite standard and follows from the fact that from the Reflection Rule one can prove uni-

queness of elimination for arbitrary eliminators for non-circular inductive sets.
The proof idea is as follows. Let f and g be functions which are defined on an inductive set A and assume that their results are judgementally equal when applied to objects in set A which are in constructor form. Then by elimination over the inductive set under consideration one can prove that f and g when applied to arbitary objects x in set A give results f(x) and g(x) which are equal in the sense of propositional equality. Now by the Reflection Rule one can conclude that f(x) and g(x) are judgementally equal for arbitrary objects x in the set A. From this we can conclude that [x : A] f(x) and [x : A] g(x) are judgementally equal and by the η-rule of the logical framework we finally get that f and g are judgementally equal.

Thus we have shown that - suprisingly - the *Reflection Rule can be expressed in a purely equational way* by an η-like for the eliminator J. We know that in case of the type-free or the simply typed λ-calculus the η-rule does not destroy the Church-Rosser property of the rewrite system providing the operational semantics of these functional languages. Therefore, at first sight, one might imagine that one can give a well-behaved operational semantics for the underlying functional language of intensional set theory as well because we only add something like an η-rule to the rewrite system. On the other hand we have seen that some computation steps in extensional set theory require proving arbitrarily complex equality propositions. In principle it might be the case that although some computation steps are very complex adding the η-like rule for J nevertheless gives rise to a well-behaved operational semantics. But this is not the case as the η-like rule for J leads to a non-confluent term rewriting system for the underlying functional language.

This can be seen from the following counterexample.

Consider again the function

$$\text{pr1} := [x : A]\,[y : A]\,[z : \text{Id}\ A\ x\ y]\ x \in \{x : A\}\ \{y : A\}\ \{z : \text{Id}\ A\ x\ y\}\ A$$

from the proof of Theorem 1.1 . Then due to the η-like rule for J

$$\text{J}\ A\ C\ ([x : A]\ e\ x\ x\ (r\ A\ x)) \Longrightarrow e$$

and by putting $C := [x : A]\,[y : A]\,[z : \text{Id}\ A\ x\ y]\ A$ and $e := \text{pr1}$ we get

$$\text{J}\ A\ C\ ([x : A]\ \text{pr1}\ x\ x\ (r\ A\ x)) \Longrightarrow \text{pr1}$$

Due to the definition of pr1 we get by applying β-reduction to the redex pr1 x x (r A x) that

$$\text{J}\ A\ C\ ([x : A]\ \text{pr1}\ x\ x\ (r\ A\ x)) \Longrightarrow \text{J}\ A\ C\ ([x : A]\ x)$$

But, obviously, pr1 , i.e. [x : A] [y : A] [z : Id A x y] x , and J A C ([x : A] x) are different normal forms.

An alternative - maybe easier - way of showing that adding the η-like rule for J to the rewrite system does not lead to a confluent and strongly normalizing rewrite system is the following. If one adds the η-like rule for J then one can derive the Reflection Rule which in turn allows to derive

$$\text{pr1} = \text{pr2} \in \{x : A\}\ \{y : A\}\ \{z : \text{Id}\ A\ x\ y\}\ A$$

Therefore if the extended term rewriting system were confluent the terms pr1 and pr2 would have to reduce to a common normal form. But both pr1 and pr2 themselves are already in normal form and different.

Thus there is no chance to add the η-like rule for J to the rewrite system and to keep at the same time the meta-mathematical property of confluence. For the same reason it is impossible to keep the even stronger property of strong normalization which is badly needed when one wants to make type checking decidable.

The conclusion we have to draw is that adding η-like rules to a rewrite system for a functional language is very problematic. It only works well for the classical case of dependent product types.
Therefore we are forced to accept that there is a principal difference between β–like and η–like rules. The β–like rules are necessary for computation and therefore harmless whereas the η–like rules in almost all cases lead to a non-confluent rewrite system, i.e. to a very badly behaved, non-deterministic notion of computation.
On the other hand η–like rules allow to prove the *judgemental* equality of functions defined over a non-circular inductive set simply by checking that they behave the same way on objects in constructor form or - in case of real circularity of the inductive definition - that they satisfy the same recursion scheme. Therefore if one insists on the identification of propositional and judgemental equality there is no way to avoid these η–like rules and therefore one gets as a consequence that one cannot interpret proofs as programs because the notion of computation for proofs is not well behaved - namely highly nondeterministic.

Thus if one wants to keep the paradigm of proofs-as-programs one has to give up the reflection rule and stick to intensional type theory.

Of course, now the problem arises to *understand* the difference between judgemental and propositional equality - a subtle distinction not very common-place even in constructive mathematics.

For a semantically oriented person this, of course, can be achieved in the best way by considering models where these two different notions of equality do not coincide, i.e. to study *non-extensional models of constructive set theory.* The investigation of such models will be a central subject of this work and will be treated in the subsequent Chapters 2 and 3 .
But before we want to improve the intuitive understanding of the difference between judgemental and propositional equality.

The best way to understand it - I believe - is to consider the identity type $\mathrm{Id}\ A\ t\ s$ as a proposition stating a relation between the *objects denoted by the terms* t and s, respectively, whereas the judgement $t = s \in A$ is a statement of a relation between the *terms* t and s, namely that one can prove their equality by purely equational reasoning using only the rules of the logical framework and the β-like rules for the eliminators.
Now the point of view of intensional constructive set theory is that closed terms which are well formed, i.e. have a type, *denote equal objects iff the terms are convertible*. That surely is evident for simple basic sets such as natural numbers, enumeration types, lists, trees etc., *but* for higher types as e.g. $N \to N$ it is a topic of debate whether functions should be considered as equal only if they are convertible. E.g. if we consider the functions

$$t \equiv [x{:}N]\ x \qquad \text{and} \qquad s \equiv [x{:}N]\ R\ 0\ ([x{:}N][y{:}N]\ \mathrm{succ}\ y)\ x$$

of type $N \to N$ of our original example which both compute the identity function on natural numbers then it might seem hard to accept that they are different. On the other hand when considered as algorithms they are different as their time complexity is different (constant vs. linear !).
Anyway in intensional type theory the situation is that the judgement $t = s \in N \to N$ is not provable and there cannot be a term p of type $\mathrm{Id}\ (N{\to}N)\ t\ s$ as this would by the Identity Reflection Lemma entail the judgement $t = s \in N \to N$.
Nevertheless *it is also impossible to derive the negation of the proposition* $\mathrm{Id}\ (N{\to}N)\ t\ s$, i.e. to exhibit a term of type $(\mathrm{Id}\ (N{\to}N)\ t\ s) \to N_0$. The reason is that there is a model, namely the naive set-theoretical one, where t and s get the same interpretation and therefore the type $(\mathrm{Id}\ (N{\to}N)\ t\ s) \to N_0$ is empty. Similarly this holds for the various realizability models based on ω-Set as studied e.g. in [Str1] and more generally in any model of extensional CST which is consistent in the sense that N_0 is empty.
Obviously, as intensional CST simply is a restriction of extensional CST less sequents become provable in the intensional variant. Therefore it might be better to use the more negative word "non-extensional" instead of the more positive word "intensional" which misleadingly suggest that one can speak about intensional aspects of objects which cannot be expressed in the extensional theory. As we do not want to violate a well established terminology we nevertheless stick to the word "intensional" although we only mean "non-extensional".
One might call a CST *strongly intensional* if it would allow to derive some proposition as inhabited which are empty in some model of extensional set theory. E.g. one might postulate

some constant p of type $(\mathrm{Id}\ (N \to N)\ t\ s) \to N_0$ which in extensional CST - where one can find an object q of type $\mathrm{Id}\ (N \to N)\ t\ s$ - would have the consequence that $p(q) \in N_0$.
This surely could be done in a consistent way but it seems to be impossible to find intuitive and correct computation rules for the constant p . Therefore - if one wants to do this at all - it would be sufficient to reason relative to a context where p is declared as a variable of type $(\mathrm{Id}\ (N \to N)\ t\ s) \to N_0$.
Anyway, it does not seem very worthwhile to have a version of CST where one can prove that functional objects are intensionally different although one can also prove that they are pointwise equal (as e.g. for our terms t and s where one can find a term of the propositional type $(\mathrm{Id}\ (N \to N)\ t\ s) \to N_0$ as one can find a term of type $t \in \Pi\ N\ ([n{:}N]\ \mathrm{Id}\ N\ (t\ n)\ (s\ n)\ )$. The reason is that the notion of intensional or judgemental equality is heavily syntax-dependent and therefore *ad hoc* , i.e. one might think of an alternative syntactic representation system for abstract objects like functions or simply of additional conversion rules where terms which previously were not convertible get convertible by adding new computation rules.
But surely it is a question of philosophical debate whether one wants to make a distinction between a term and the object it denotes. The point is that the assumption of the existence of abstract mathematical objects (as e.g. functions) independent from syntax may be considered as a quite idealistic attitude. Nevertheless, at least for people trained in Tarskian model-theoretic semantics the distinction between syntactic representations and the abstract objects they denote is very helpful in organizing their thoughts.

Summing up the considerations above the situation to our opinion is the following.
Even if one would like to speak about extensional equality in the framework of Martin-Löf's constructive set theory it is not possible to do so as long as one wants to keep meta-mathematical properties of the underlying functional language such as confluence and strong normalisation. As long as one wants to extract programs from proofs one cannot give up such constraints as well-behavedness of the operational semantics of the (functional) programs.
This leads to the conclusion that *one cannot identify programs and constructive proofs to such an extent as intended originally without either extending ones notion of functional program in an unreasonable way or restricting ones notion of mathematics to a non-extensional one.*
Intensional constructive set theory seems to be of great importance, especially from a practical point of view, but it is not yet very well understood from a semantical point of view. Therefore we are mainly interested in understanding intensional constructive set theory by studying models which refute a lot of judgements for which heuristic considerations have suggested that they are not derivable in intensional constructive set theory.

But before starting semantics we want to expand on some considerations about the strength of current formulations of intensional constructive set theory thereby suggesting and studying a natural and quite powerful new eliminator for identity sets.

1.2 Limitations of Current Formulations of Intensional CST and How to Avoid Them by the New Eliminator K

Quite generally in constructive set theory a basic assumption is that any object of an inductive set is "equal" to an object in constructor form, i.e. objects of dependent product sets are functional abstractions, objects of sum sets are pairs, objects of identity sets are of the form r A a etc. But, of course, there immediately arises the question which sense of equality is meant. This question is relevant w.r.t. nonempty contexts only as otherwise both notions of equality do coincide.
Later on we shall see that there are models where not any typable term (in a non-empty context) is judgementally equal to a term in constructor form. This is due to the fact that in these models uniqueness of elimination is not valid in the sense of judgemental equality. Nevertheless we will show that in the sense of propositional equality any term is equal to a term in constructor form by proving that uniqueness of elimination holds in the sense of propositional equality.

For illustrating the general scheme we first will construct a term p proving that *any object of product type is propositionally equal to an object in constructor form*, i.e.

A : Set , B : {x : A} Set , c : Π A B
⊢
p ∈ Id (Π A B) c (λ A B ([x : A] apply A B c x)) .

Due to F-elimination it is enough to prove the propositional equality for most general objects in constructor form of the type Π A B .

From the conversion rule for F we immediately get the derived conversion rule for apply :

A : Set , B : {x : A} Set , f : {x : A} B x , x : A
⊢
apply A B (λ A B f) x = f x ∈ B x

Therefore we get by functional abstraction that

A : Set , B : {x : A} Set , f : {x : A} B x
⊢
[x : A] apply A B (λ A B f) x = [x : A] f x ∈ {x : A} B x

As η-conversion holds on the level of the logical framework we further get that

A : Set , B : {A} Set , f : {x : A} B x

$\vdash$
[x : A] f x = f ∈ {x : A} B x

and by equality resoning we then get

A : Set , B : {x : A} Set , f : {x : A} B x
$\vdash$
f = [x : A] apply A B (λ A B f) ∈ {x : A} B x

Further by applying λ-abstraction we get

A : Set , B : {x : A} Set , f : {x : A} B x
$\vdash$
(λ A B f) = λ A B ([x : A] apply A B (λ A B f) x) ∈ Π A B

From this and the following instance of the introduction rule for identity sets

A : Set , B : {x : A} Set , f : {x : A} B x
$\vdash$
r (Π A B) (λ A B f) ∈ Id (Π A B) (λ A B f) (λ A B f)

we get that

A : Set , B : {x : A} Set , f : {x : A} B x
$\vdash$
r (Π A B) (λ A B f) ∈ Id (Π A B) (λ A B f) (λ A B ([x : A] apply A B (λ A B f) x))

Finally by F-elimination we get

A : Set , B : {x : A} Set , c : Π A B
$\vdash$
F A B ([c : Π A B] Id (Π A B) c (λ A B ([x : A] apply A B c x)))
    ([f : {x : A} B x] r (Π A B) (λ A B f))
    c
∈
Id (Π A B) c (λ A B ([x : A] apply A B c x))

Following the same pattern of proof we can also show that any object of a sum type is propositionally equal to an object in constructor form , i.e. that there is a term p with

$$A : \mathrm{Set}\,,\ B : \{x : A\}\ \mathrm{Set}\,,\ c : \mathrm{El}(\Sigma\ A\ B)$$
$$\vdash$$
$$p \in \mathrm{Id}\ (\Sigma\ A\ B)\ c\ (\mathrm{pair}\ A\ B\ (\pi_0\ A\ B\ c)\ (\pi_1\ A\ B\ c))$$

where π_0 and π_1 are the projections on the first and second component, respectively, and are defined as follows

$$\pi_0 \equiv [A : \mathrm{Set}]\ [B : \{x : A\}\ \mathrm{Set}]\ [c : \Sigma\ A\ B]$$
$$E\ A\ B\ ([z : \Sigma\ A\ B]\ A)\ ([x : A]\ [y : B\ x]\ x)\ c$$

of type $[A : \mathrm{Set}\}\ \{B : \{x : A\}\ \mathrm{Set}\}\ \{c : \Sigma\ A\ B\}\ A$ and

$$\pi_1 \equiv [A : \mathrm{Set}]\ [B : \{x : A\}\ \mathrm{Set}]\ [c : \Sigma\ A\ B]$$
$$E\ A\ B\ ([z : \Sigma\ A\ B]\ B\ (\pi_0\ A\ B\ z))\ ([x : A]\ [y : B\ x]\ y)\ c$$

of type $[A : \mathrm{Set}\}\ \{B : \{x : A\}\ \mathrm{Set}\}\ \{c : \Sigma\ A\ B\}\ B(\pi_0\ A\ B\ c)$.

For these projection operations one can derive the sequents

$$A : \mathrm{Set}\,,\ B : \{x : A\}\ \mathrm{Set}\ ,\ a : A\,,\ b : B\ a\ \vdash\ \pi_0\ A\ B(\mathrm{pair}\ A\ B\ a\ b) = a \in A$$

$$A : \mathrm{Set}\,,\ B : \{x : A\}\mathrm{Set}\ ,\ a : A\,,\ b : B\ a\ \vdash\ \pi_1 A\ B(\mathrm{pair}\ A\ B\ a\ b) = b \in B\ a$$

from which we get that

$$A : \mathrm{Set}\,,\ B : \{x : A\}\ \mathrm{Set}\ ,\ a : A\,,\ b : B\ a$$
$$\vdash$$
$$\mathrm{pair}\ A\ B\ a\ b = \mathrm{pair}\ A\ B\ (\pi_0\ A\ B(\mathrm{pair}\ A\ B\ a\ b))\ (\pi_1\ A\ B(\mathrm{pair}\ A\ B\ a\ b)) \in \Sigma\ A\ B$$

From this and the following instance of the Id-introduction rule

$$A : \mathrm{Set}\,,\ B : \{x : A\}\ \mathrm{Set}\ ,\ a : A\,,\ b : B\ a$$
$$\vdash$$
$$r\ (\Sigma\ A\ B)\ (\mathrm{pair}\ A\ B\ a\ b) \in \mathrm{Id}\ (\Sigma\ A\ B)\ (\mathrm{pair}\ A\ B\ a\ b)\ (\mathrm{pair}\ A\ B\ a\ b)$$

we get

$$A : \mathrm{Set}\,,\ B : \{x : A\}\ \mathrm{Set}\ ,\ a : A\,,\ b : B\ a$$
$$\vdash$$
$$r\ (\Sigma\ A\ B)\ (\mathrm{pair}\ A\ B\ a\ b)$$

$\in$
Id (Σ A B) (pair A B a b) (pair A B (π_0 A B(pair A B a b)) (π_1 A B(pair A B a b)))

Finally by E-elimination we get

A : Set , B : {x : A} Set , c : Σ A B
$\vdash$
E A B
([c : Σ A B] Id (Σ A B) c (pair A B (π_0 A B c) (π_1 A B c)))
([a : A] [b : B a] r (Σ A B) (pair A B a b))
c
$\in$
Id (Σ A B) c (pair A B (π_0 A B c) (π_0 A B c))

Of course, one would like to derive a similar result for identity sets. But in this case the question is a bit more involved as Id is not simply an inductive set but an *inductive family of sets* in the sense of [Dyb] . In the presence of sums of families of sets a good approximation to an inductively defined family can be obtained by replacing the family by its (iterated) sum. This can be done quite generally but we will need this only for the special, but typical, case of the inductively defined family of identity sets.

Instead of the family of sets Id $\in$ [x : A] [y : A] Set we consider the set

A_S $\equiv$ Σ A ([x : A] Σ A ([y : A] Id A x y))

representing the iterated disjoint union of Id and instead of r the function we consider

r_S $\equiv$ [x : A] triple A x x (r A x)

where

triple $\equiv$ [A : Set] [x : A] [y : A] [z : Id A x y]
pair A ([x : A] Σ A ([y : A] Id A x y)) x (pair A ([y : A] Id A x y) y z)

Now we will show that A_S together with the constructor r_S carries an inductive structure.

Let C $\in$ {t : A_S} Set and d $\in$ {x : A} C (r_S x) then the term

t1 $\equiv$ J A ([x : A][y : A][z : Id A x y] C (triple A x y z)) d x y z satisfies the sequent

$$x : A ,\ y : A ,\ z : \mathrm{Id}\ A\ x\ y \vdash t1 \in C\ (\mathrm{triple}\ A\ x\ y\ z)$$

and the term

$$\begin{aligned} t2 \equiv E\ A\ & ([y : A]\ \mathrm{Id}\ A\ x\ y)\ ([z : \Sigma\ A\ ([y : A]\ \mathrm{Id}\ A\ x\ y)]\ C\ (\mathrm{triple}\ A\ x\ y\ z)) \\ & ([y : A][z : \mathrm{Id}\ A\ x\ y]\ t1) \\ & v \end{aligned}$$

satisfies the sequent

$$x : A ,\ v : \Sigma\ A\ ([y : A]\ \mathrm{Id}\ A\ x\ y) \vdash t2 \in C\ (\mathrm{pair}\ A\ ([y : A]\ \mathrm{Id}\ A\ x\ y)\ x\ v)$$

Then the term

$$\begin{aligned} t \equiv E\ A\ & ([x : A]\ \Sigma\ A\ ([y : A]\ \mathrm{Id}\ A\ x\ y))\ C \\ & ([x : A]\ [v : \Sigma\ A\ ([y : A]\ \mathrm{Id}\ A\ x\ y)]\ t2) \end{aligned}$$

satisfies the sequent

$$u : \Sigma\ A\ ([x : A]\ \Sigma\ A\ ([y : A]\ \mathrm{Id}\ A\ x\ y)) \vdash t\ u \in C\ u$$

i.e.

$$u : A_S \vdash t\ u \in C\ u$$

and by straightforward application of the reduction rules we get that

$$x : A \vdash t\ (r_S\ x) = d\ x \in C\ (r_S\ x)$$

Next we will derive that for the inductive type A_S every object $c \in A_S$ is propositionally equal to $r_S\ t$ for some term t of type A. It is sufficient to show that for arbitrary $x : A$, $y : A$, $z : \mathrm{Id}\ A\ x\ y$ the objects $\mathrm{triple}\ x\ y\ z$ and $r_S\ x$ of type A_S are propositionally equal. This propositional equality is exhibited by the term

$$p \equiv J\ A\ ([x : A][y : A][z : \mathrm{Id}\ A\ x\ y]\ \mathrm{Id}\ A_S\ (\mathrm{triple}\ A\ x\ y\ z)\ \ (r_S\ x))\ ([x : A]\ r\ A_S\ (r_S\ x))$$

which is of type

$$\{x : A\}\ \{y : A\}\ \{z : \mathrm{Id}\ A\ x\ y\}\ \ \mathrm{Id}\ A_S\ (\mathrm{triple}\ A\ x\ y\ z)\ (r_S\ x)$$

as the term $[x : A]\ r\ A_S\ (r_S\ x)$ is of type $\{x : A\}\ \mathrm{Id}\ A_S\ (r_S\ x)\ (r_S\ x)$ which is definitionally equal to $\{x : A\}\ \mathrm{Id}\ A_S\ (\mathrm{triple}\ A\ x\ x\ (r\ A\ x))\ (r_S\ x)$.

By applying E-elimination twice one can finally get from p a proof term q such that

$$c : A_S \vdash q\ c \in \mathrm{Id}\ A_S\ c\ (r_S\ (\pi_0\ A\ ([x : A]\ \Sigma\ A\ ([y : A]\ \mathrm{Id}\ A\ x\ y\ ))\ c))$$

After having shown that relative to the context $x : A$, $y : A$, $z : \mathrm{Id}\ A\ x\ y$ the objects triple x y z and triple x x (r A x) of type A_S are propositionally equal one might think that it is easy to show that also z and r A x are propositionally equal. But this is impossible as the terms z and r A x are of types Id A x y and Id A x x , respectively, which are not convertible because x and y are not convertible ! From the point of view of intensional constructive set theory this has to be accepted as quite natural as it seems to be an intrinsic feature of intensionality that Id A x y and Id A x x are not judgementally equal and therefore neither the type Id (Id A x y) z (r A x) nor the type Id (Id A x x) z (r A x) are well defined.
But under the stronger assumption that the object z is of type Id A x x one might hope to show that z is propositionally equal to the object r A x . This is a meaningful question as - due to the assumption about z - both r A x and z are terms of the same type Id A x x .
Intuitively, this claim should be valid for intensional type theory as for any variable free type expression A , any term t of type A and any term p of type Id A t t - due to strong normalization - the term p reduces to a term of the form r A' t' where A converts to A' and t converts to t' .

Thus one would expect that in intensional type theory one can derive the corresponding proposition, i.e. find a term of type {x : A} {z : Id A x x} Id (Id A x x) z (r A x) .
Unfortunately, this seems to be impossible ! We will discuss several attempts of finding a term of type {x : A} {z : Id A x x} Id (Id A x x) z (r A x) *and explain why they fail.*
Furthermore we shall introduce a new eliminator K *for* Id-types *which appears quite naturally and solves the problem.*

A first, most natural attempt would be to define a C of type {x : A}{y : A}{z : Id A x y} Set s.t. [x : A][z : Id A x x] C x x (r A x) converts to [x : A][z : Id A x x] Id (Id A x x) z (r A x) .
In that case the term

$$[x : A]\ [z : \mathrm{Id}\ A\ x\ x]\ \ J\ A\ C\ ([x : A]\ r\ (\mathrm{Id}\ A\ x\ x)\ (r\ A\ x))\ x\ x\ z$$

would be of type {x : A} {z : Id A x x} Id (Id A x x) z (r A x) but unfortunately such a C does not seem to exist as the most immediate candidate

$$[x : A]\ [y : A]\ [z : \mathrm{Id}\ A\ x\ y]\ \mathrm{Id}\ (\mathrm{Id}\ A\ x\ x)\ z\ (r\ A\ x)$$

is not typable as z is of type Id A x y which does not convert to the expected type Id A x y .

Another attempt would be to prove the propositional equality of z and r A x from the propositional equality of triple A x x z and triple A x x (r A x) . But in order to perform this step we would have to rely on the validity of the sequent

A : Set , B : {A} Set ,
a : A , b1 : B a , b2 : B a , c : Id (Σ A B) (pair A B a b1) (pair A B a b2)
⊢
p ∈ Id (B a) b1 b2

whose derivation - which will be given later on as the proof of Theorem 1.5 - essentially makes use of the eliminator K which allows to solve the original problem more directly.

But before we will go to technicalities let us discuss what is the impact of this apparent incompleteness of the current formulation of intensional constructive set theory.
It is not surprising that there must appear incompleteness phenomena as - due to Gödel's Incompleteness Theorem - for any nontrivial formal system containing arithmetic there must be a *gap between formal provability and validity.* But what *is* surprising is that the apparently underivable proposition {x : A} {z : Id A x x} Id (Id A x x) z (r A x) is of such a simple form and so evident intuitively.
A possible "philosophical" explanation seems to be that the proposition under consideration states the propositional equality of objects which usually do not belong to the ontology of even constructive mathematics - namely of proof objects. In everyday mathematics such objects appear as meta-objects about which nothing can be expressed inside the object language. That is not the case for type theory as there one does not speak *about formal proofs but about proof objects*. But as proof objects are a kind of "quasi-meta-objects" which allow to speak about some meta-aspects in the object language - necessarily in an approximative way in order to keep consistency - they are very likely to cause incompleteness phenomena which is a behaviour typical for meta-statements.

Now one might think that this is an over-interpretation as in its purest form the kind of incompleteness we are discussing shows up already in a formulation where the proposition which seems to be unprovable does not at all refer to proof objects : find a proof term p such that

A : Set , B : {A} Set ,
a : A , b1 : B a , b2 : B a , c : Id (Σ A B) (pair A B a b1) (pair A B a b2)
⊢
p ∈ Id (B a) b1 b2

Here the proposition Id (B a) b1 b2 does not refer to any proof object - nevertheless it seems to be impossible to prove the proposition as we do not know how to construct from proof object c - establishing that the objects (pair A B a b1) and (pair A B a b2) are propositionally equal - a proof object establishing that their second projections are propositionally equal, too.

Nevertheless, I do not think that the "philosophical" discussion above is an over-interpretation because finding a proof term p with

a : A , b1 : B a , b2 : B a ,
c : Id (Σ A B) (pair A B a b1) (pair A B a b2)
$\vdash$
p ∈ Id (B a) b1 b2

is a problem *only if the family* B is *not constant* . But the only way of introducing non-constant families of sets is by using Identity Types !

Therefore - I think - one can draw the conclusion that incompleteness phenomena for propositions of simple form do arise due to the fact that ordinary mathematical types and propositional types - i.e. types of "meta-objects" - are not carefully separated and live in the same conceptual world.
But as we think that this mixture of data types and propositional types - though a bit strange from the point of view of main stream mathematics - is very appealing and challenging we shall propose a new construct which allows to cope with the incompleteness phenomena discussed above.

The very reason for these apparent incompleteness phenomena is that we lack a concept which allows us to prove in a straightforward way that objects z and r A x of type Id A x x are propositionally equal relative to the context x : A , z : Id A x x .

The usual way to express in type-theoretic language that for any set A and any object a of type A the type Id A a a contains exactly the object r A a is to claim that for any family of sets C ∈ {x : A} {z : Id A x x} Set and d ∈ {x : A} C x (r A x) there is a natural extension of d to an f ∈ {x : A}{z : Id A x x} C x z with f a (r A a) = d a ∈ C a (r A a) for all a ∈ A. To express that this natural extension can be chosen uniformly in A , C and d we denote it by K A C d f .
The intuition behind this formulation is the following. If for some object a ∈ A there were an object c ∈ Id A a a different from r A a then for any family C ∈ {x : A}{z : Id A x x} Set with C a c empty there could not exist an f of type {x : A} {z : Id A x x} C x z even if there were an object d ∈ {x:A} C x (r A x) as then we would have f a c ∈ C a c contradicting the

assumption that C a c is empty. But, of course, the family C with C a (r A a) the singleton type and C a c the empty type for objects c of type Id A a a different from r A a is such a family and nevertheless due to the assumption there is an f ∈ {x : A} {z : Id A x x} C x z as there is a function d ∈ {x : A} C x (r A x) with d a the unique object of the singleton type for any a ∈ A . Thus there cannot exist an object c of an identity type Id A a a such that c is different from r A a .

More formally, the rules governing the use of K can be stated in a way following the well known pattern for elimination operators.

Definition 1.2

There is a constant

K ∈ {A : Set} {C : {x : A} {z : Id A x x} Set} {d : {x : A} C x (r A x)}
{x : A} {z : Id A x x} C x z

satisfying the conversion rule

A : Set , C : {x : A} {z : Id A x x} Set , d : {x : A} C x (r A x) , a : A
⊢
K A C d a (r A a) = d a ∈ C a (r A a) ◊

Using the terminology of inductive types, see [Dyb] , one can rephrase the definition above informally as follows. For any set A not only the family of sets Id A ∈ {a , b : A} Set carries an inductive structure (expressed by the eliminator J) but also the subfamily [a : A] Id A a a carries an inductive structure as given by the constructor r A and the eliminator K .
Compared with the traditional approach to inductive types it seems a bit unorthodox to endow one and the same family of sets with two different inductive structures. But isn't it the case that identity sets themselves are a little bit unorthodox ?!

Next we will show that using our new eliminator K it is quite easy to prove that any object of an identity type Id A a a is propositionally equal to the object r A a in constructor form.

Theorem 1.3

Let C ≡ [A : Set] [x : A] [z : Id A x x] Id (Id A x x) z (r A x) then for the term

prf1 ≡ [A : Set] K A (C A) ([x : A] r (Id A x x) (r A x))

it holds that

prf1 ∈ {A : Set}{x : A}{z : Id A x x} Id (Id A x x) z (r A x)

Furthermore a K-eliminator can be defined from any hypothetical term

prf ∈ {A : Set}{x : A}{z : Id A x x} Id (Id A x x) z (r A x)

satisfying the conversion rule

A : Set , a : A
⊢
prf A a (r A a) = r (Id A a a) (r A a) ∈ Id (Id A a a) (r A a) (r A a)

Proof : First we construct an appropriate term prf1 using K . As

A : Set ⊢ [x : A] r (Id A x x) (r A x) ∈ {x : A} C A (r A x) (r A x)

we have due to the definition of K that the term

prf1 ≡ [A : Set] K A (C A) ([x : A] r (Id A x x) (r A x))

is of type

{A : Set} {x : A}{z : Id A x x} C A z (r A x)

which by expanding the definition of C converts to

{A : Set} {x : A}{z : Id A x x} Id (Id A x x) z (r A x)

For proving the second part of the theorem assume that

prf ∈ {A : Set}{x : A}{z : Id A x x} Id (Id A x x) z (r A x)

satisfying the conversion rule

A : Set , a : A
⊢
prf A a (r A a) = r (Id A a a) (r A a) ∈ Id (Id A a a) (r A a) (r A a)

First we define the auxiliary terms sub and sym (see alo Appendix).

The term

```
aux ≡

    [A : Set] [B : {x : A} Set]
       J  A
          ([x : A] [y : A] [z : Id A x y] Π (B x) ([u : B x] B y))
          ([x : A] fun (B x) ([u : B x] B x) ([u : B x] u))
```

is of type

```
{A : Set} {B : {x : A} Set} {x : A} {y : A} {z : Id A x y} Π (B x) ([u : B x] B y)
```

and therefore the term

```
sub ≡

    [A : Set] [B : {x : A} Set] [x : A] [y : A] [z : Id A x y] [u : B x]
       apply (B x) ([u : B x] B y) (aux A B x y z) u
```

is of type

```
    {A : Set} {B : {x : A} Set} {x : A} {y : A} {z : Id A x y} {u : B x} B y
```

and from the conversion rules it follows that

```
    A : Set , B : {x : A} Set , x : A , u : B x
    ⊢
    sub A B x x (r A x) u  =  u ∈  B x
```

The term

```
sym ≡ [A : Set] J A ([x : A] [y : A] [z : Id A x y] Id A y x) ([p : Id A x x] p)
```

is of type

```
       {A : Set} {x : A} {y : A} {z : Id A x y} Id A y x
```

and from the conversion rules it follows that

A : Set , a : A ⊢ sym A x x (r A x) = r A x ∈ Id A x x

Then the term

```
K1 ≡

   [A : Set] [C : {x : A} {z : Id A x x} Set] [d : {x : A} C x (r A x)]
   [a : A] [c : Id A a a]
      sub (Id A a a) (C a) (r A a) c (sym A a a (prf A a c)) (d a)
```

is of type

```
{A : Set} {C : {x : A} {z : Id A x x} Set} {d : {x : A} C x (r A x)}
{a : A} {c : Id A a a}
   C a c
```

and by expanding definitions and applying conversion rules we get that

```
A : Set , C : {x : A} {z : Id A x x} Set , d : {x : A} C x (r A x) , a : A
⊢
K1 A C d a (r A a) = d a ∈ C a c
```

Therefore K1 satisfies the characterizing properties of the K-eliminator . ◊

Remark An immediate, but important consequence of the previous theorem is that we can define the term

```
irrel1 ≡

   [A:Set] [B : {x:A} Set]

      K A ([a:A] [c: Id A a a] Π (B a) ([b : B a] Id (B a) (sub A B a a c b) b))

      ([a:A] fun (B a)

                ([b : B a] Id (B a) (sub A B a a (r A a) b) b)

                ([b : B a] r (B a) b))
```

with

irrel1 $\in$ {A : Set}{B : {x:A} Set} {a:A} {c : Id A a a}

Π (B a) ([b : B a] Id (B a) (sub A B a a c b) b)

expressing that sub A B a a c b is propositionally equal to b for any c $\in$ Id A a a .
But, of course, sub A B a a c b and b are not judgementally equal unless c is judgementally equal to r A a .

Next we will show that using the eliminator K one can prove that all objects of an identity type are propositionally equal.

Theorem 1.4

There is a term prf2 with

prf2 $\in$ {A : Set}{x : A}{y : A}{u : Id A a b}{v : Id A x y} Id (Id A x y) u v

Furthermore employing prf2 there is a term

irrel $\in$

{A : Set}{B : {x:A} Set} {a1,a2 : A} {b : B a1} {p,q : Id A a 1 a2}

Id (B a2) (sub A B a1 a2 p b) (sub A B a1 a2 q b)

establishing that the result of substituting along the proof of some propositional equality is independent - up to propositional equality - from the choice of this proof object.

Proof : For the term (where prf1 is as in Theorem 1.3)

prf_aux $\equiv$

J A
([x : A][y : A][z : Id A x y] Π (Id A x y) ([u : Id A x y] Id (Id A x y) u z))
([x : A] fun (Id A x x) ([u : Id A x x] Id (Id A x x) u (r A x)) ([u : Id A x x] prf1 A x u))

we have that

```
A : Set
⊢
prf_aux ∈ {x : A}{y : A}{u : Id A x y} Π (Id A x y) ([v : Id A a b] Id (Id A x y) v u)
```

due to the definition of the eliminator J as the term

```
[x : A] fun (Id A x x) ([u : Id A x x] Id (Id A x x) u (r A x)) ([u : Id A x x] prf1 A x u)
```

is of type

```
{x : A} Π (Id A x x) ([u : Id A x x] Id (Id A x x) u (r A x))
```

Therefore the term

```
prf2 ≡ [A : Set] [x : A] [y : A] [u : Id A x y] [v : Id A x y]
        apply (Id A x y) ([u : Id A x y] Id (Id A x y) u v) (prf_aux A x y v) u
```

is of type

```
{A : Set}{x : A}{y : A}{u : Id A x y}{v : Id A x y} Id (Id A x y) u v
```

For the second part of the theorem consider the term

```
irrel ≡

   [A : Set] [B : {x:A} Set] [a1,a2 : A] [b : B a1] [p,q : Id A a 1 a2]

      J  (Id A a 1 a2)

         ([p,q : Id A a 1 a2] [u : Id (Id A a 1 a2) p q]

               Id (B a2) (sub A B a1 a2 p b) (sub A B a1 a2 q b))

         ([p : Id A a 1 a2] r (B a2) (sub A B a1 a2 p b))

         p q

         (prf2 A a1 a2 p q)
```

which is of the desired type

{A : Set}{B : {x:A} Set} {a1,a2 : A} {b : B a1} {p,q : Id A a 1 a2}

Id (B a2) (sub A B a1 a2 p b) (sub A B a1 a2 q b)

as

[p : Id A a 1 a2] r (B a2) (sub A B a1 a2 p b)

is of the type

Id (B a2) (sub A B a1 a2 p b) (sub A B a1 a2 q b) [p , p, r (Id A a 1 a2) p / p , q , u]

and

(prf2 A a1 a2 p q) $\in$ Id (Id A a 1 a2) p q

◊

Next we show that the eliminator K allows to prove a theorem which allows to derive (without using K) the propositional equality of z and r A x from the propositional equality of triple A x x z and triple A x x (r A x) .

Theorem 1.5

Assuming K one can construct a term

prf3 $\in$ {A : Set} {B : {x : A} Set}{a : A} {b1 : B a}{b2 : B a}
{p : Id (Σ A B) (pair A B a b1) (pair A B a b2)}
Id (B a) b1 b2

Proof : First we consider the term

eq_fst $\equiv$ [A : Set] [B : {x : A} Set]
J (Σ A B)
([u : Σ A B] [v : Σ A B] [w : Id (Σ A B) u v] Id A (π_0 A B u) (π_0 A B v))
([u : Σ A B] r A (π_0 A B u))

of type

{A : Set} {B : {x : A} Set}
{u : Σ A B} {v : Σ A B} {w : Id (Σ A B) u v} Id A (π_0 A B u) (π_0 A B v)

and the term

```
eq_snd  ≡

[A : Set] [B : {x : A} Set]
[u : Σ A B] [v : Σ A B] [w : Id (Σ A B) u v]
    J  (Σ A B)
       ( [u : Σ A B] [v : Σ A B] [w : Id (Σ A B) u v]
          Id (B (π_0 A B v))
             (sub A B (π_0 A B u) (π_0 A B v) (eq_fst A B u v w) (π_1 A B u))
             (π_1 A B v) )
       ( [u : Σ A B] r (B (π_0 A B u)) (π_1 A B u) )
```

of type

```
    {A : Set} {B : {x : A} Set} {u : Σ A B} {v : Σ A B} {w : Id (Σ A B) u v}
       Id (B (π_0 A B v))
          (sub A B (π_0 A B u) (π_0 A B v) (eq_fst A B u v w) (π_1 A B u))
          (π_1 A B v) )
```

The term

```
aux  ≡

[A : Set] [B : {x : A} Set]
[a : A] [b1 : B a] [b2 : B a] [p : Id (Σ A B) (pair A B a b1) (pair A B a b2)]
   eq_snd A B (pair A B a b1) (pair A B a b2) p
```

is of type

```
{A : Set} {B : {x : A} Set}
{a : A} {b1 : B a} {b2 : B a} {p : Id (Σ A B) (pair A B a b1) (pair A B a b2)}
   Id (B a) (sub A B a a (eq_fst A B (pair A B a b1) (pair A B a b2) p) b1) b2
```

Then the term (where prf1 is the term constructed in the proof of Theorem 1.3)

```
prf3  ≡

   [A : Set] [B : {x : A} Set]
```

```
[a : A] [b1 : B a] [b2 : B a] [p : Id (Σ A B) (pair A B a b1) (pair A B a b2)]
    sub
        (Id A a a)
        ([q : Id A a a] Id (B a) (sub A B a a q b1) b2)
        (eq_fst A B (pair A B a b1) (pair A B a b2) p)
        (r (Id A a a) (r A a))
        (prf1 A a (eq_fst A B (pair A B a b1) (pair A B a b2) p))
        (aux A B a b1 b2 p)
```

is of type

```
{A : Set} {B : {x : A} Set}
{a : A} {b1 : B a} {b2 : B a} {p : Id (Σ A B) (pair A B a b1) (pair A B a b2)}
  Id (B a) (sub A B a a (r A a) b1) b2
```

which converts to

```
{A : Set} {B : {x : A} Set}
{a : A} {b1 : B a} {b2 : B a} {p : Id (Σ A B) (pair A B a b1) (pair A B a b2)}
  Id (B a) b1 b2
```

By lengthy, but straightforward expansion of definitions and application of conversion rules we finally get that

```
A : Set , B : {x : A} Set
a : A , b : B a
⊢
prf3 A B a b b (r (Σ A B) (pair A B a b)) = r (B a) b ∈ Id (B a) b b    ◊
```

The theorems above - I think - have demonstrated that it is worthwhile to add an elimination operator K guaranteeing that - up to propositional equality - any type of the form Id A a a contains exactly the object r A a .

We will subsequently give an alternative, but equivalent axiomatization of identity types using the eliminators id_elim_ess and id_elim_uni which express the *essence of identity* and the *uniqueness of proof objects of identity type*, respectively.

Definition 1.6

Suppose that we have

id_elim_ess $\in$ {A : Set}{C : {x : A}{y : A} Set}{d : {x : A} C x x}
{a : A}{b : A}{c : Id A a b} C a b

together with the conversion rule

A : Set , C : {x : A}{y : A} Set , d : {x : A} C x x , a : A
$\vdash$
id_elim_ess A C d a a (r A a) = d a $\in$ C a a

and

id_elim_uni $\in$ {A : Set} {a : A}{C : {z : Id A a a} Set}{d : C (r A a)}
{c : Id A a a} C c

together with the conversion rule

A : Set , a : A , C : {z : Id A a a} Set , d : C (r A a)
$\vdash$
id_elim_uni A a C d (r A a) = d $\in$ C (r A a) $\Diamond$

We will now show that this new alternative definition of identity types is equivalent to the eliminators J and K .

Theorem 1.7

The eliminators id_elim_ess and id_elim–uni can be defined in terms of J and K and vice versa.

Proof : The eliminator id_elim_ess can be defined from the eliminator J as

[A : Set] [C : {x : A}{y : A} Set] [d : {x : A} C x x]
J A ([x : A][y : A][z : Id A x y] C x y) d

Next we define the eliminator id_elim_ess from the eliminators J and K in several steps.

First notice that the term

aux $\equiv$

[A : Set] [a : A] [C : {z : Id A a a} Set] [d : C (r A a)]

```
    J (Id A a a)
      ([u : Id A a a][v : Id A a a][p : Id (Id A a a) u v] Π (C v) ([x : C v] C u))
      ([u : Id A a a] fun (C u) ([x : C u] C u) ([x : C u] x))
```

is of type

```
  {A : Set} {a : A} {C : {z : Id A a a} Set} {d : C (r A a)}
  {u : Id A a a} {v : Id A a a} {p : Id (Id A a a) u v}
  Π (C v) ([x : C v] C u)
```

Recall that the term

```
  prf1  ≡  [A : Set]  K A (C A) ([x : A] r (Id A x x) (r A x))
```

from Theorem 1.3 is of type

```
  {A : Set}{x : A}{z : Id A x x} Id (Id A x x) z (r A x)
```

Then the term

```
 [A : Set] [a : A] [C : {z : Id A a a} Set] [d : C (r A a)] [c : Id A a a]
     aux A a C d c (r A a) (prf1  A a c)
```

is of type

```
  {A : Set} {a : A} {C : {z : Id A a a} Set} {d : C (r A a)}
    {c : Id A a a} Π (C (r A a)) ([x : C (r A a)] C c)
```

But then the term

```
  t1  ≡

  [A : Set] [a : A] [C : {z : Id A a a} Set] [d : C (r A a)] [c : Id A a a]
     apply (C (r A a)) ([x : C (r A a)] C c) (aux A a C d c (r A a) (prf1  A a c)) d
```

is of type

```
  {A : Set} {a : A} {C : {z : Id A a a} Set} {d : C (r A a)} {c : Id A a a} C c
```

and using the definitions and the conversion rules one gets that

```
  A : Set , a : A , C : {z : Id A a a} Set , d : C (r A a)
```

```
    ⊢
    t1 A a C d (r A a) = d ∈ C (r A a)
```

Therefore the term t1 satisfies the defining conditions for id_elim_ess .

Next we show how to define J from id_elim_ess and id_elim_uni in several steps.

First notice that

```
    A : Set , C : {x : A} {y : A} {z : Id A x y} Set , d : {x : A} C x x (r A x) , x : A
    ⊢
    id_elim_uni A x (C x x) (d x) ∈ {z : Id A x x} C x x z
```

and therefore the term

```
  [A : Set] [C : {x : A} {y : A} {z : Id A x y} Set] [d : {x : A} C x x (r A x)]
    id_elim_ess A ([x : A] [y : A] Π (Id A x y) ([z : Id A x y] C x y z))
      ( [x : A]
         fun (Id A x x) ([z : Id A x x] C x x z) (id_elim_uni A x (C x x) (d x)) )
```

is of type

```
    {A : Set} {C : {x : A} {y : A} {z : Id A x y} Set} {d : {x : A} C x x (r A x)}
    {x : A} {y : A} {z : Id A x y}
      Π (Id A x y) ([z : Id A x y] C x y z)
```

Therefore the term

```
  t1 ≡

  [A : Set] [C : {x : A} {y : A} {z : Id A x y} Set] [d : {x : A} C x x (r A x)]
    apply
    (Id A x y) ([z : Id A x y] C x y z)
    (id_elim_ess A ([x : A] [y : A] Π (Id A x y) ([z : Id A x y] C x y z))
      ( [x : A]
         fun (Id A x x) ([z : Id A x x] C x x z) (id_elim_uni A x (C x x) (d x))))
```

is of type

```
    {A : Set} {C : {x : A} {y : A} {z : Id A x y} Set} {d : {x : A} C x x (r A x)}
    {x : A} {y : A} {z : Id A x y} C x y z)
```

Straightforward equality reasoning shows that

A : Set , C : {x : A} {y : A} {z : Id A x y} Set , d : {x : A} C x x (r A x) , x : A
$\vdash$
t2 A C d a a (r A a) = d $\in$ C a a (r A a)

Thus the term t2 fulfills the defining conditions of J .

Finally, the eliminator K can be defined from the eliminator id_elim_uni simply as

[A : Set] [C : {x : A}{z : Id A x x}] [d : {x : A} C x (r A x)]
[a : A] id_elim_uni A a (C a) (d a) $\Diamond$

The only difference of id_elim_ess to the traditional elimination operator J is that the family C must not depend on a proof object of identity type. This definitely is no real restriction in the practice of formalizing *constructive mathematics* as there *one usually does not speak about proof objects.*
The elimination operator id_elem_ess is a type-theoretic version of the following proof rule characterizing equality in e.g. first order logic

$$\vdash P(x, x) \quad \text{iff} \quad x = y \vdash P(x, y) \quad .$$

This proof rule may seem somewhat unconventional but it directly corresponds to the categorical explanation for the semantics of equality as given in the framework of hyperdoctrines by F. W. Lawvere, see [Law1]. He explains equality on type A as the predicate $\exists_\delta(\mathrm{true}_A)$ where $\delta : A \to A \times A$ is the diagonal morphism and $\exists_\delta$ is the left adjoint to the reindexing functor δ^* .

Intuitively, the eliminator id_elim_uni expresses that an identity type Id A a a contains exactly the object r A a . Suppose that there were an object c in Id A a a different from the canonical object r A a then one could construct an object of the empty type N_0 as follows : let C be the family of sets indexed over Id A a a with C (r A a) = N_1 and C c = N_0 then id_elim_uni A a C n_0 c $\in$ N_0 .
In some sense it is conceptually clearer to base identity types on the elimination operators id_elim_ess and id_elim_uni as the first eliminator expresses exactly the *logical essence of equality* as needed for formalizing constructive mathematics whereas the second eliminator is just used for proving propositions expressing properties of proof objects such as the proposition *there is at most one proof object for identity types*. This kind of propositions is not really part of constructive mathematics but belongs to a kind of restricted meta-reasoning which is possible in constructive set theory but not in traditional logical systems.

The reason why we nevertheless use the equivalent formulation based on J and K is that this definition fits more nicely into the scheme for defining elimination operations for (families of) inductive sets as exposed in [Dyb] .

We finish this section by proving that the eliminator K is definable from the eliminator J in intensional constructive set theories where for all $A \in$ Set and families $B \in A \rightarrow$ Set if a1 and a2 are propositionally equal objects of set A then B a1 and B a2 are already judgementally equal as sets.
This latter condition is satisfied for the type theory studied by Jan Smith in his Thesis, see [Smi] . Models validating this condition will be studied in Chapter 2 of the present work.

Theorem 1.9

Assuming the sequent

A : Set , B : {a : A} Set , a1 : A , a2 : A , b : B a1 , p : Id A a1 a2
⊢
B a1 = B a2 ∈ Set

one can define the eliminator K in terms of the eliminator J .

Proof : From the assumption it follows immediately that

A : Set , a1 : A , a2 : A , p : Id A a1 a2 ⊢ Id A a1 a1 = Id A a1 a2 ∈ Set

and therefore

A : Set , a1 : A , a2 : A , p : Id A a1 a2 ⊢ p ∈ Id A a1 a1

From this it follows that

A : Set , C : {a : A} {c : Id A a a} Set
a1 : A , a2 : A , p : Id A a1 a2
⊢
C a1 p ∈ Set

and therefore we get that

A : Set , C : {a : A} {c : Id A a a} Set
⊢

```
[a1 : A] [a2 : A] [p : Id A a1 a2] C a1 p
∈
{a1 : A} {a2 : A} {p : Id A a1 a2} Set
```

From this and from

```
A : Set , C : {a : A} {c : Id A a a} Set , d : {a : A} C a (r A a)
⊢
d ∈ ([a1 : A] [a2 : A] [p : Id A a1 a2] C a1 p) a a (r A a)
```

it follows due to the definition of J that

```
A : Set , C : {a : A} {c : Id A a a} Set , d : {a : A} C a (r A a)
⊢
J A ([a1 : A] [a2 : A] [p : Id A a1 a2] C a1 p) d
∈
{a1 : A} {a2 : A} {p : Id A a1 a2} C a1 p
```

But from this we get

```
A : Set , C : {a : A} {c : Id A a a} Set , d : {a : A} C a (r A a) ,
a : A , c : Id A a a
⊢
J A ([a1 : A] [a2 : A] [p : Id A a1 a2] C a1 p) d a a c ∈ C a c
```

and therefore the term

```
K0 ≡ [A : Set] [C : {a : A] [c : Id A a a} Set] [d : {a : A} C a (r A a)]
     [a : A] [c : Id A a a]
        J A ([a1 : A] [a2 : A] [p : Id A a1 a2] C a1 p) d a a c
```

is of the intended type

```
{A : Set} {C : {a : A} {c : Id A a a} Set} {d : {a : A} C a (r A a)}
   {a : A} {c : Id A a a} C a c
```

and due to the conversion rule for J we also have

```
A : Set , C : {a : A} {c : Id A a a} Set , d : {a : A} C a (r A a) , a : A
⊢
J A ([a1 : A] [a2 : A] [p : Id A a1 a2] C a1 p) d a a (r A a) = d a ∈ C a (r A a)
```

i.e. we have

A : Set , C : {a : A} {c : Id A a a} Set , d : {a : A} C a (r A a) , a : A
⊢
K_0 A C d a a (r A a) = d a ∈ C a (r A a)

Therefore the term K_0 satisfies all the defining conditions for the eliminator K ...... ◊

1.3 Paulin-Mohring's Eliminator for Identity Types

As a further application of the eliminator K we show that it allows to define another eliminator for identity types which originally has been introduced by Ch. Paulin-Mohring. In deriving Paulin-Mohring's eliminator it is very helpful to rely on the auxiliry eliminator id_elim_uni which has been defined in terms of J and K in the previous Theorem 1.7 .
Afterwards we prove a theorem due to Th. Altenkirch and H. Goguen telling that Paulin-Mohring's eliminator is definable from J only provided that Set is a universe closed under impredicative universal quantification.
Paulin-Mohring's eliminator is the canonical eliminator for the family

Id ∈ {A : Set} {a : A} {b : A} Set

when A ∈ Set and a ∈ A are considered as *parameters* and b is considered as an index in the sense of [Dyb] .

Theorem 1.8

Using the eliminators id_elim_ess and id_elim_uni or equivalently using the eliminators J and K one can define a term PM with

PM ∈ {A : Set} {a : A} {C : {y : A}{z : Id A a y} Set} {d : C a (r A a)}
{b : A} {c : Id A a b} C b c

satisfying the conversion law

A : Set , a : A , C : {y : A} {z : Id A a y} Set , d : C a (r A a)
⊢
PM A a C d a (r A a) = d ∈ C a (r A a)

Proof : The term

```
aux1 ≡

    [A : Set] [a : A] [C : {y : A} {z : Id A a y} Set] [d : C a (r A a)]
       id_elim_uni A a ([z : Id A a a] C a z) d
```

is of type

```
    {A : Set} {a : A} {C : {y : A} {z : Id A a y} Set} {d : C a (r A a)}
       {c : Id A a a} C a c
```

The term

```
aux2 ≡

    [A : Set] [a : A] [C : {y : A} {z : Id A a y} Set] [d : C a (r A a)]
       fun (Id A a a) ([z : Id A a a] C a z) (aux1 A a C d)
```

is of type

```
    {A : Set} {a : A} {C : {y : A} {z : Id A a y} Set} {d : C a (r A a)}
       Π (Id A a a) ([z : Id A a a] C a z)
```

Then the term

```
aux3 ≡

[A : Set] [a : A] [C : {y : A} {z : Id A a y} Set] [d : C a (r A a)]
[b : A] [c : Id A b]
      sub A ([y : A] Π (Id A a a) ([z : Id A a y] C a z)) a b c (aux2 A a C d))
```

is of type

```
{A : Set} {a : A} {C : {y : A} {z : Id A a y} Set} {d : C a (r A a)
{b : A} {c : Id A b}
      Π (Id A a a) ([z : Id A a a] C a z) → Π (Id A a b) ([z : Id A a b] C a z)
```

and the term

```
aux4 ≡

[A : Set] [a : A] [C : {y : A} {z : Id A a y} Set] [d : C a (r A a)]
```

```
 [b : A] [c : Id A b]
     apply
        (Π (Id A a a) ([z : Id A a a] C a z))
        ([f : Π (Id A a a) ([z : Id A a a] C a z)] Π (Id A a b) ([z : Id A a b] C a z))
        (aux3 A a C d b c)
        (aux2 A a C d)
```

is of type

```
 {A : Set} {a : A} {C : {y : A} {z : Id A a y} Set} {d :  C a (r A a)
 {b : A} {c : Id A b}
    Π (Id A a b) ([z : Id A a b] C a z)
```

Finally we get the term

```
 PM  ≡

 [A : Set] [a : A] [C : {y : A} {z : Id A a y} Set] [d :  C a (r A a)]
 [b : A] [c : Id A b]
    apply (Id A a b) ([z : Id A a b] C a z) (aux4 A a C d b c) c
```

of type

```
 {A : Set} {a : A} {C : {y : A} {z : Id A a y} Set} {d :  C a (r A a)
    {b : A} {c : Id A b}
       C b c
```

and straightforward, but tedious application of conversion rules gives

```
    A : Set , a : A , C : {y : A} {z : Id A a y} Set , d :  C a (r A a)
    ⊢
    PM A a C d a (r A a)  =  d  ∈  C b c                              ◊
```

It is remarkable that the eliminator PM is definable from J without using K if one assumes that Set is closed under products of arbitrary families of sets, i.e. if the product of a family of sets is a set even if the index set is not a set but an arbitrary type.

Theorem 1.9 (Th. Altenkirch, H. Goguen)

If Set is closed under products of arbitrary families of sets not necessarily indexed by a set then the eliminator PM is definable in terms of J without using K .

Proof : To facilitate reading we employ the following conventions. We write $(\Pi\ x : A)\ B[x]$ for $\Pi\ A\ ([x : A]\ B[x])$, $(\lambda\ x : A)\ t$ for fun A $([x : A]\ B[x])\ ([x : A]\ t)$ (provided that t is of type $B[x]$ if x is of type A) and finally we simply write f a for apply A $([x : A]\ B[x])$ f a (provided that f is of type $\Pi\ A\ ([x : A]\ B[x])$) .

The term

```
help ≡

[A : Set]
   J  A
   ( [x : A] [y : A] [z : Id A x y] (Π C : (Id A x y) → Set) (C x (r A x)) → C y z) )
   ( [x : A] (λ C : (Id A x y) → Set) (λ p : C x (r A x) p )
```

is of type

```
{A : Set} {x : A} {y : A} {z : Id A x y} (Π C : (Id A x y) → Set) (C x (r A x)) → C y z)
```

and therefore the term

```
t ≡  [A : Set] [x : A] [C : (Id A x y) → Set] [d : C x (r A x)]
     [y : A] [z : Id A x y]
        help A x y z C d
```

is of type

```
{A : Set} {x : A} {C : (Id A x y) → Set} {d : C x (r A x)}
{y : A} {z : Id A x y}
   C y z
```

By straightforward application of conversion rules one gets the conversion law

```
A : Set , x : A , C : (Id A x y) → Set , d : C x (r A x)
⊢
t A a x C d x (r A x)  =  d ∈  C x (r A x)
```

Thus the term t fullfills the defining properties of PM . ◊

1.4 Identity Sets for Contexts

Finally we want to define a notion of identity set for *arbitrary set contexts* instead of single sets only . For these generalized identity sets we define eliminators generalizing the eliminators J and K .

More precisely, for any natural number n with identity sets for set contexts of length n there are associated n+1 eliminators $J_{n,i}$ (i=0,...n).

In the presence of Σ-types it is sufficient to use only Id_1 , r_1 , $J_{1,1}$ and $J_{1,0}$ which correspond to the usual Id , r , J and K studied above.

Definition 1.10

For any natural number n there is a type constructor

$$\begin{aligned} Id_n \in \; & \{A_1 : Set\}\, \{A_2 : A_1 \to Set\} \ldots \{A_n : \{x_1 : A_1\} \ldots \{x_n : A_n\, x_1 \ldots x_{n-1}\}\, Set\} \\ & \{x_1 : A_1\}\, \{x_1 : A_1\} \ldots \{x_n : A_n\, x_1 \ldots x_{n-1}\}\, \{y_n : A_n\, y_1 \ldots y_{n-1}\} \\ & \quad Set \end{aligned}$$

and a constructor

$r_n \in$

$$\begin{aligned} & \{A_1 : Set\}\, \{A_2 : A_1 \to Set\} \ldots \{A_n : \{x_1 : A_1\} \ldots \{x_n : A_n\, x_1 \ldots x_{n-1}\}\, Set\} \\ & \{x_1 : A_1\} \ldots \{x_n : A_n\, x_1 \ldots x_{n-1}\} \\ & \quad Id\, A_1 \ldots A_n\, x_1\, x_1 \ldots x_n\, x_n \end{aligned}$$

and for any natural number i with $0 \leq i \leq n$ there is an eliminator $J_{n,i}$ (see Table 1) together with the conversion rule

$$J_{n,i}\, A_1 \ldots A_n\, C\, d\, x_1 \ldots x_i\, x_{i+1}\, x_{i+1} \ldots x_n\, x_n\, (r\, A_1 \ldots A_n\, x_1 \ldots x_n) = d\, x_1 \ldots x_n$$

$J_{n,i} \in$

$\{A_1 : \mathrm{Set}\}\ \{A_2 : A_1 \to \mathrm{Set}\}\ \ldots\ \{A_n : \{x_1 : A_1\}\ \ldots\ \{x_n : A_n\, x_1 \ldots x_{n-1}\}\ \mathrm{Set}\}$
$\{C\ :\ \{x_1 : A_1\}$

$\quad .$

$\quad \{x_i : A_i\, x_1 \ldots x_{i-1}\}$
$\quad \{x_{i+1} : A_{i+1}\, x_1 \ldots x_i\}\ \{y_{i+1} : A_{i+1}\, x_1 \ldots x_i\}$

$\quad .$

$\quad \{x_n : A_n\, x_1 \ldots x_{n-1}\}\ \{y_n : A_n\, x_1 \ldots x_i\, y_{i+1} \ldots y_{n-1}\}$
$\quad \{z : \mathrm{Id}\ A_1 \ldots A_n\, x_1\, x_1 \ldots x_i\, x_i\, x_{i+1}\, y_{i+1} \ldots x_n\, y_n\}$
$\quad\quad \mathrm{Set}\ \}$
$\{d :\ \{x_1 : A_1\}\}$

$\quad .$

$\quad \{x_n : A_n\, x_1 \ldots x_{n-1}\}$
$\quad\quad C\, x_1 \ldots x_i\, x_{i+1}\, x_{i+1} \ldots x_n\, x_n\, (r\, A_1 \ldots A_n\, x_1 \ldots x_n)\ \}$
$\{x_1 : A_1\}$

$.$

$\{x_i : A_i\, x_1 \ldots x_{i-1}\}$
$\{x_{i+1} : A_{i+1}\, x_1 \ldots x_i\}\ \{y_{i+1} : A_{i+1}\, x_1 \ldots x_i\}$

$.$

$\{x_n : A_n\, x_1 \ldots x_{n-1}\}\ \{y_n : A_n\, x_1 \ldots x_i\, y_{i+1} \ldots y_{n-1}\}$
$\{z : \mathrm{Id}\ A_1 \ldots A_n\, x_1\, x_1 \ldots x_i\, x_i\, x_{i+1}\, y_{i+1} \ldots x_n\, y_n\}$
$\quad C\, x_1 \ldots x_i\, x_{i+1}\, x_{i+1} \ldots x_n\, x_n\, (r\, A_1 \ldots A_n\, x_1 \ldots x_n)$

Table 1

◊

1.5. All Proof Objects of Identity Types Are Propositionally Equal In All Models Satisfying the Uniformity Principle

We have seen previously that in the traditional formulation of ICST without K , as presented e.g. in [NPS], it seems very unlikely that one can find a term which provably is of type

UNIQUE ≡ {A:Set}{a:A} {p : Id A a a} Id (Id A a a) p (r A a)

But there is a recent argument of Th. Altenkirch showing that in any reasonable model of ICST, i.e. any model satisfying the so called *Uniformity Principle*, the type UNIQUE is provably inhabited by a term. As all mathematical models we know satisfy this uniformity principle this is a further indication that refuting UNIQUE requires a subtle argument and cannot be done semantically.

We first present an argument due to Th. Altenkirch providing a tentative proof of UNIQUE.

Theorem 1.11 (Th. Altenkirch)

There are terms t1 and t2 with

t1 ∈ {A:Set}{a:A}{p : Id A a a} Id (Id A a a) (sub A ([u:A] Id A u a) a a p p) (r A a)

t2 ∈ {A:Set}{a:A}{p : Id A a a} Id (Id A a a) (sub A ([u:A] Id A a a) a a p p) p

If there *were* a term t3 for which one can derive

t3 ∈ {A:Set}{a:A}{p : Id A a a}
Id (Id A a a) (sub A ([u:A] Id A u a) a a p p) (sub A ([u:A] Id A a a) a a p p)

then one *could* construct a term t for which one can derive

t ∈ {A:Set}{a:A} {p : Id A a a} Id (Id A a a) p (r A a)

Proof : Recall that

sub ≡ [A : Set] [B : {x : A} Set] [x : A] [y : A] [z : Id A x y] [u : B x]
apply (B x) ([u : B x] B y) (aux A B x y z) u

where

```
aux  ≡ [A : Set] [B : {x : A} Set]
          J  A
             ([x : A] [y : A] [z : Id A x y] Π (B x) ([u : B x] B y))
             ([x : A] fun (B x) ([u : B x] B x) ([u : B x] u))
```

Then for the term

```
s1  ≡ {A:Set}  J A
                ([x:A][y:A][z : Id A x y]
                   Id (Id A y y) (sub A ([u:A] Id A u y) x y z z) (r A y))
                ([x:A] r (Id A x x) (r A x))
```

we have

$$s1 \in \{A{:}Set\}\{a{:}A\}\{b{:}A\}\{p : Id\ A\ a\ b\}\ Id\ (Id\ A\ b\ b)\ (sub\ A\ ([u{:}A]\ Id\ A\ u\ b)\ a\ b\ p\ p)\ (r\ A\ b)$$

as it holds that

$$A : Set\,,\ x : A\ \vdash\ sub\ A\ ([u{:}A]\ Id\ A\ u\ x)\ x\ x\ (r\ A\ x)\ (r\ A\ x)) \ =\ r\ A\ x\ \in\ Id\ A\ x\ x$$

and for the term

```
s2  ≡ {A:Set}  J A
                ([x:A][y:A][z : Id A x y]
                   Id (Id A y y) (sub A ([u:A] Id A x y) x y z z) z)
                ([x:A] r (Id A x x) (r A x))
```

we have

$$s2 \in \{A{:}Set\}\{a{:}A\}\{b{:}A\}\{p : Id\ A\ a\ b\}\ Id\ (Id\ A\ a\ b)\ (sub\ A\ ([u{:}A]\ Id\ A\ a\ b)\ a\ b\ p\ p)\ p$$

as it holds that

$$A : Set\,,\ x : A\ \vdash\ sub\ A\ ([u{:}A]\ Id\ A\ x\ x)\ x\ x\ (r\ A\ x)\ (r\ A\ x)) \ =\ r\ A\ x\ \in\ Id\ A\ x\ x$$

Now by putting

```
t1  ≡ [A:Set] [a:A] [p : Id A a a]  s1 A a a p

t2  ≡ [A:Set] [a:A] [p : Id A a a]  s2 A a a p
```

we get that

t1 ∈ {A:Set}{a:A}{p : Id A a a} Id (Id A a a) (sub A ([u:A] Id A u a) a a p p) (r A a)

t2 ∈ {A:Set}{a:A}{p : Id A a a} Id (Id A a a) (sub A ([u:A] Id A a a) a a p p) p

Now if there were a term t3 for which one can derive

t3 ∈ {A:Set}{a:A}{p : Id A a a}
Id (Id A a a) (sub A ([u:A] Id A u a) a a p p) (sub A ([u:A] Id A a a) a a p p)

then simply using symmetry and transitivity of propositional equality one could construct from t1 , t2 , t3 a term t of type {A:Set}{a:A}{p : Id A a a} Id (Id A a a) p (r A a) .

◊

The importance of Altenkirch's observation is that it reduces the derivability of inhabitedness of UNIQUE to the derivability of

A:Set , a:A , p : Id A a a
⊢
Id (Id A a a) (sub A ([u:A] Id A u a) a a p p) (sub A ([u:A] Id A a a) a a p p)

Now, although the formal derivation of this sequent seems to require again our new eliminator K it is nevertheless the case that the terms

sub A ([u:A] Id A u b) a b p p and sub A ([u:A] Id A a b) a b p p

are almost convertible. They are only different w.r.t. the second argument of sub which is of type {x:A} Set .

Now it is obvious that any algorithm which can be expressed by a term in ICST does not depend on those arguments which are either sets or families of sets. The reason is that these types do not carry an inductive structure. This suggests the following new proof rule

Uniformity Principle

$$\frac{\Gamma \vdash t \in A \qquad \Gamma \vdash t \in A \qquad \mathrm{strip}[t] \equiv \mathrm{strip}[s] \in A}{\Gamma \vdash t = s \in A}$$

where for any term t the term strip[t] is obtained from t by throwing away all subterms which are either sets or families of sets.

The next theorem tells that assuming the Uniformity principle on can derive UNIQUE.

Theorem 1.11

Under the assumption of the Uniformity Principle one can derive

```
    A : Set , a : A , p : Id A a a
    ⊢
    sub A ([u:A] Id A u a) a a p p  =  sub A ([u:A] Id A a a) a a p p  ∈  Id A a a
```

and therefore find a term which provably is of type

```
    UNIQUE  ≡  {A:Set}{a:A} {p : Id A a a} Id (Id A a a) p (r A a)
```

Proof : Due to the Uniformity Principle it is sufficient to show that

```
    strip[sub A ([u:A] Id A u a) a a p p]  ≡  strip[sub A ([u:A] Id A a a) a a p p]
```

Unfolding the definitions of sub and aux we get that

```
  sub A ([u:A] Id A u a) a a p p  ≡

  apply (Id A a a) ([u : Id A a a] Id A u a)
      (J  A
        ([x : A] [y : A] [z : Id A x y] Π (Id A x a) ([u : Id A x a] Id A y a))
        ([x : A] fun (Id A x a) ([u : Id A x a] Id A x a) ([u : Id A x a] u))
        a a p)
        p
```

and that

```
  sub A ([u:A] Id A a a) a a p p  ≡

  apply (Id A a a) ([u : Id A a a] Id A a a)
      (J  A
        ([x : A] [y : A] [z : Id A x y] Π (Id A a a) ([u : Id A a a] Id A a a))
        ([x : A] fun (Id A a a) ([u : Id A a a] Id A a a) ([u : Id A a a] u))
        a a p)
        p
```

But in *both* cases by stripping we obtain the same term

```
    apply (J ([x] fun ([u] u)) a a p) p          ◊
```

One should remark here that in a formulation of ICST where already in the definition of terms sets or families of sets cannot appear as subterms the Uniformity Principle is automatically valid. Such a formulation has been called "polymorphic" in [NPS] as opposed to the formulation inside a logical framework, which we have employed and in [NPS] is called "monomorphic".
Furthermore in all the models we know the Uniformity Principle is semantically valid. The reason is that it seems rather impossible to endow the type Set with a non-uniform structure. Typically in the case of realizability models the type Set always carries the trivial realizability structure (any number realizes any set !).
Our impression is that models where the type Set carries a nontrivial realizability structure one would have to introduce *codes* for sets and then one gets structures which are already as complicated as term models !
Therefore we think that any proof showing that the type UNIQUE *is not inhabited must be of syntactical nature !*

1.6. Conclusion

Finishing our syntactic considerations we would like to say as a conclusion that the *current formulation of intensional constructive set theory seems to be open to extensions*. When dealing with specific problems one has to extend the theory by a new construct as e.g. our new eliminator K (which has been invented independently from us by Th. Altenkirch in Edinburgh). For other applications other new extensions might be useful. A quite powerful extension of the definition mechanism has been described by Th. Coquand in [Coq] and has been implemented in ALF.
In this paper there are studied general schemes for defining functions by *complete* and *non-overlapping* case analysis on the possible constructor forms of the arguments of the function to be defined. This general definition scheme together with a check for termination has been implemented in the ALF system developped at the University of Gotenburg.
We think that any *natural model* (e.g. those studied in Chapter 2 and 3) *of intensional constructive set theory must be a model also for all of these extensions.* The use of these models should be to use them as a test tool in the sense that when one wants to introduce a new concept one first should check whether it can be interpreted in one of these models.

Chapter 2

Semi-Intensional Models of Constructive Set Theory

In this chapter we introduce and discuss the first non-extensional model of ICST. It satisfies most of the requirements for intensionality with one single exception discussed in section 6 . Nevertheless it is a model for a rather established semi-intensional type theory studied by J. Smith in his Thesis (cf. [Smi]).
In the section 1 we recall the basics of the semantics of extensional type theory as studied e.g. in [Str1]. Based on this material we construct a first non-extensional model which is inadequate in the sense that all sets are inhabited (i.e. we have logical inconsistency).
In section 2 we give a motivation for the crucial distinction between potential and actual realizers which allows to distinguish between intensional and extensional equality of functions and originates back to Kreisel's notion of modified realizability.
In section 3 we describe the category IRS of intensional realizability sets obtained from an arbitrary category of domains by the process of *sconing* which in category theory is also known as *Freyd cover* of a category or more generally as *glueing* (cf. [FS],[LS]).
In section 4 we show that from the category IRS one can construct a contextual category with dependent products (cf. [Str1]) serving as a model for the ambient logical framework.
In section 5 we show how to interpret the type Set and the usual concepts of ICST such as Π, Σ , Id , N , N_0 , N_1 etc. and the associated constructors and eliminators. An important point is that our new eliminator K can be interpreted straightforwardly in a way quite similar to J .
In section 6 we show the first independence results for ICST as e.g. the non-validity of η-rules and of the extensionality principle for (number-theoretic) functions. Thus the models of this chapter are intensional in many aspects. But we show that, nevertheless, they are only semi-intensional as for a family of sets B indexed over a set A the sets B(t) and B(s) are already judgementally equal even if t and s are only propositionally equal w.r.t. A .
In section 7 we summarize our discussion of the models based on sconings of categories of domains and conclude that they are suitable for refuting sequents which are derivable in extensional, but not in intensional CST.

2.1 Semantical Background and a First Simple Non-Extensional Model

In the last years there has emerged a lot of work on the semantics of Martin-Löf's *extensional constructive set theory*. A detailed account of one of these approaches, namely contextual categories, can be found e.g. in the author's book [Str1] together with a discussion of related work.

In this and the next chapter we definitely want to build on this work and extend it to get an improved semantic understanding of Martin-Löf's *intensional constructive set theory* (ICST). First we will recapitulate the definition of semantics for ICST which - as far as the formulation *without* a logical framework is concerned - can be found already in Cartmell's Thesis [Cart]. Anyway, to defining the *semantics* of ICST, i.e. to define an appropriate *notion of model* for ICST, requires only a few modifications of the work in [Str1].
The notion of a *contextual category with products of families of types* as defined in [Str1] and originally due to J. Cartmell, is the adequate notion of semantics for LF, the Logical Framework, see also [Pym]. Remember that at this level there is no need to worry about intensionality as for the Logical Framework one assumes that the η-rule for functional abstraction is valid.
We know that ICST appears as a *special theory in the Logical Framework* : there is a type constant Set together with a generic family of types El and a lot of other constants for the usual type and object forming operations and for the elimination operations.
Therefore we will not define formally what our notion of model for ICST is as there is hardly anything new in it. Obviously, a model for ICST is a *contextual category C with products of families of types* together with an interpretation of Set , El and the constants for the usual type and object forming operations and for the elimination operations such that w.r.t. the interpretation in C all the laws of ICST (see Appendix) are valid.
Instead we concentrate on the *construction of specific models* for ICST which are natural in the sense that they exhibit a rich mathematical structure *but nevertheless are not models of extensional constructive set theory,* i.e. do *not validate the η–like rules for the elimination operations and satisfy the three criteria for intensionality given below.*
But - as all interesting models - they will validate much more sequents than are formally provable in ICST. Typically they allow to interpret the new eliminator K which has been discussed at length in Chapter 1 . We consider this as a justification of this new eliminator. In the same way these models can be used as a first semantical check whether an attempt of introducing an extension of ICST is consistent. Of course, if one finds that the suggested extension can be interpreted in one of the models this does not yet guarantee that this extension is also *computationally meaningful* !
For extensional constructive set theory *realizability models* have proved to be useful if not even to be the intended models. We will use slight modifications of these realizability models in order to obtain *intensional models of* ICST , i.e. models which do not validate the η-like rules typical for extensional constructive set theory.
First we consider a model for intensional constructive set theory which is obtained from the well known ω-Set model by interpreting Set by a suitable subcollection of PERω .We assume that the underlying coding has a special behaviour w.r.t. 0 , namely that $<0\,,0> = 0$ and $\{0\}(n) = 0$ for all $n \in \omega$. Let PER-0 denote the set of all partial equivalence relations on ω with $0 \mathrel{R} 0$.
The logical framework is interpreted in the category ω-Set of ω-sets and realizable morphisms. That this constitutes a model of extensional dependently typed λ-calculus has been

shown in [Str1].
The type Set is interpreted as the ω-set with a trivial realizability relation whose underlying collection of objects is PER-0 . For any object R ∈ Set , i.e. for any partial equivalence relation R on ω with 0 R 0 , its associated type El(R) will be interpreted as the ω-set whose underlying set of elements is ω/R , the set of equivalence classes modulo R , and whose realizability relation is simply membership, i.e. n realizes A ∈ ω/R iff n ∈ A . In [Str2] it has been shown that PER-0 is closed under products of families indexed over *arbitrary* ω-sets and strong sums of families which are *indexed over ω-sets originating from pers in* PER-0 .
In the paper [Str2] it has been shown that in this model it is impossible to interpret extensional identity types. The reason simply is that any type of the from El(R) is nonempty and therefore it is impossible to interpret Id A a b if a and b are different objects of type A .
But we will see that it is possible to interpret intensional identity sets in this model in a rather simple way. Let A be an arbitary type, i.e. ω-set, and a , b ∈ A then Id A a b will be interpreted by the per { (0 , 0) , (1 , 1) } if a = b and as the set { (0 , 0) } if a ≠ b . For any object a ∈ A the object r A a will be interpreted as the object {1} ∈ El (Id A a a). Notice that any identity set contains the object {0} playing the role of a *kind of "error element"*.
For defining theinnterpretation of the elimination operation J we shall use the fact that for any A ∈ Set , i.e. any A ∈ PER-0 , the ω-set corresponding to El(A) contains a dummy or default object realized by 0 , namely the equivalence class of 0 modulo R .
Now if C is an object of the type {x : A} {y : El A} {z : Id A x y} Set and d is an object of {x : A} C x x (r A x) and a and b are objects of type A and c is an object of Id A a b then we define J A C d a b c as follows :

(i) if c is realized by 0 then we put

J A C d a b c = the unique element of C a b c realized by 0

(ii) if c is realized by 1 then a = b and c = r A a and then we put

J A C d a b c = d a ∈ C a b c

The function J is realizable as there is an algorithm checking whether the realizer for c is 0 and then gives the realizer 0 as result and otherwise applies the realizer for d to the realizer for a .
It is obvious from this definition that for all A , C , d , a of appropriate type the model validates that J A C d a a (r A a) = d a .
But the η-like rule for J is not valid in this model as for functions f , g of type {x : A} {y : El A} {z : Id A x y} Set with f a a (r A a) = g a a (r A a) for all a ∈ A it does not necessarily hold that f = g as f and g may give different results when applied to arguments which are not of constructor form.
This is demonstrated by the following counterexample. Take for A the type N of natural

numbers, i.e. the ω-set whose underlying set is ω and for which n realizes m iff $n = m$, and for C the constant family with value N. Let f and g be the functions corresponding to the projection on the first and second argument, respectively, i.e. $f\,x\,y\,z = x$ and $g\,x\,y\,z = y$ for all $x, y \in N$ and $z \in \mathrm{Id}\,N\,x\,y$. The functions f and g are obviously different as N contains more than one element *but nevertheless* it holds that $f\,x\,x\,(r\,N\,x) = x = g\,x\,x\,(r\,N\,x)$ for all $x \in N$.
Thus the following sequent is not valid in the model under consideration

$$A : \mathrm{Set}, x : A, y : A, z : \mathrm{Id}\,A\,x\,y \vdash x = y \in A$$

But also the sequent

$$A : \mathrm{Set}, B : A \to \mathrm{Set}, x : A, y : A, z : \mathrm{Id}\,A\,x\,y \vdash x = y \in B(x) = B(y) \in \mathrm{Set}$$

is not valid as if for A we take the type N of natural numbers and for B we take the family $[x : N]\,\mathrm{Id}\,N\,0\,x$ then $\mathrm{Id}\,N\,0\,0$ and $\mathrm{Id}\,N\,0\,1$ are different objects of type Set although there is an object of type $\mathrm{Id}\,N\,0\,1$, namely the default object realized by 0.

Thus this model is very intensional as it violates the axioms of extensional type theory in a rather strong sense : *relative to a nontrivial context propositionally equal objects need not be judgementally equal and the elements of a family of types need not be equal even if their indices are propositionally equal.* This is very pleasing also as it shows that the η-like rules for Id-types - typical for extensional type theory - cannot be derived from the rules of intensional type theory in some miraculous way.

So we could be quite happy but there is something very unpleasant about the model discussed above : *any identity type is nonempty* as it is inhabited at least by the object realized by 0. This means that for all sets A and all objects $a, b \in A$ - even if a and b are different - the set $\mathrm{Id}\,A\,a\,b$ is inhabited, i.e. *true according to the paradigm of propositions as types*. But this contradicts the Equality Reflection Principle discussed in Chapter 1.

Therefore we will next study models of ICST where the Equality Reflection Principle is valid. Alas, these models will validate the sequent

$$A : \mathrm{Set}, B : A \to \mathrm{Set}, x : A, y : A, z : \mathrm{Id}\,A\,x\,y \vdash x = y \in B(x) = B(y) \in \mathrm{Set}$$

It will take until Chapter 3 that we will see models satisfying the Equality Reflection Principle *and* refuting the sequents

$$A : \mathrm{Set}, x : A, y : A, z : \mathrm{Id}\,A\,x\,y \vdash x = y \in A$$

and

$$A : \mathrm{Set}\,,\ B : A \to \mathrm{Set}\,,\ x : A\,,\ y : A\,,\ z : \mathrm{Id}\ A\ x\ y\ \vdash x = y\ \in B(x)\ =\ B(y) \in \mathrm{Set}$$

2.2 The Intuition Behind Intensional Realizability Sets

The models for ICST of this chapter will also be based on the idea of realizability - but a slightly refined one. We will not allow that an object has several different realizers or in other words : *realizers won't be forgotten but are visible ingredients of a function between intensional realizability sets.*
Technically our model constructions will be based on the notion of *Freyd cover* or *sconing* of some category of domains(cf. [FS]). The structures obtained by taking the Freyd cover of a cartesian closed category are very natural from the point of view of categorical logic which provides additional confidence in the naturality of the construction. Nevertheless this categorical aspect is not the key point to motivate these structures.

More intuitive and less technical motivation can be provided by the following purely logical considerations.
Let D be a domain theoretic model of untyped λ-calculus, i.e. that [D→D] is a retract of D maybe with some additional structure which allows to interpret extensions of the pure λ-calculus. The objects of D will serve as realizers for objects of Set . To be closer to computation one might claim that D is an effectively given domain instead of simply being a domain in the purely order theoretic sense which in general contains a lot of non-computable objects.
According to the original idea of P. Aczel, see [Acz], the sets of ICST can be conceived simply as *collections of algorithmic objects*, i.e. as subsets of D in the usual set-theoretic sense, where D appears as an *untyped universe of algorithmic objects.*
Based on this idea one obtains a model for the formulation of ICST not using an ambient Logical Framework in the following way. If A and B are subsets of D then a *map from* A *to* B simply is a domain morphism $f : D \to D$ such that $f(d) \in B$ for any $d \in A$. The crucial point now is the notion of equality for maps from A to B . Thinking extensionally, the immediate choice would be to say that maps f and g from A to B are *equal as maps from* A *to* B iff $f(d) = g(d)$ *for all* $d \in A$ *. But thinking intensionally, two maps* f *and* g *are equal iff they give the same result when applied to a most general object !* Now the idea is to replace the somewhat metaphorical, vague condition " f *and* g *give the same result when applied to a most general object*" by the more precise (according to modern set-theoretic mathematics) one that f and g give the same result when applied to *an arbitrary object* $d \in D$. Here we have replaced the vague intuitive notion of *most general object* by the mathematically precise notion of an *arbitrary object of* D which actually *is* the most general notion of algorithmic object (in our context).
Thus we have defined a non-extensional notion of equality for maps from A to B as there is

a lot of examples of maps f and g such that $f(d) = g(d)$ for all $d \in A$, i.e. f and g are *extensionally equal as maps from* A *to* B, although for some, often actually for almost all, objects $d \in D \setminus A$: $f(d) \neq g(d)$.
Nevertheless it is immediate how to define composition of maps in a way such that composition is associative : if f is a map from A to B and g is a map from B to C then $g \circ f$, the composition of f and g, is the unique map h with $h(d) = g(f(d))$ for all $d \in D$. Thus composition of maps between set types is inherited from the composition of domain morphisms from D to D which is trivially known to be associative.
If one would start from an arbitrary partial combinatory algebra (pca) as e.g. the Kleene pca then often there is no way to define composition of algorithmic operations in an associative way. But associativity is indispensible for getting categorical models and therefore we have to be more specific in our requirements for the underlying space of algorithmic objects !
Another point is that, if one looks at the formulation of ICST inside a Logical Framework, there arises the question of how to interpret types which are not sets as e.g. the type Set . According to the discussion above the underlying collection of objects for Set is $\mathcal{P}(D)$, the powerset of D, which in no way can be considered as (isomorphic to) a subset of D for reasons of cardinality.
Thus big types like Set are of quite a different nature than the small set types. Typically, in order to express *uniformity of operations which map sets to objects of sets*, one would like the type Set to a carry *a trivial realizability structure guaranteeing the desired uniformity.* That means that we would like the type Set to be represented in the following way : the underlying set of the type Set is $\mathcal{P}(D)$ and any object $A \in \mathcal{P}(D)$ is realized by the unique object of the domain 1 which is the terminal object in the category of domains under consideration.
Thus for our puposes it seems to be adequate to consider *arbitrary domains as collections of potential realizers* in contrast to the traditional work on realizability sets where a fixed partial combinatory algebra is assumed as an exhaustive collection of all possible realizers.
These considerations suggest the following general shape of what we call *intensional realizability sets* and *morphisms between intensional realizability sets.* We start from a *well behaved category of domains* refered to as **Dom** . For this one can take any of the many notions of domain currently in use or their effective versions. Then an *intensional realizability set* will consist of a *set* X of underlying objects together with a *domain* A of potential realizers and a function $r : X \rightarrow A$ associating with any object $x \in X$ its actual realizer $r(x)$ in A. Thus what is different from ordinary realizability sets such as ω-sets is that *any object comes together with its own unique distinguished realizer* and not with maybe several of them. This choice is motivated by the very idea of intensionality : maybe two different abstract objects are realized by the same algorithmic object but if abstract objects have different realizers then they must be intensionally different !
In this setting it is most natural to define a morphism from (X, A, r) to (Y, B, s) as a pair (f, a) where f is a (set-theoretic) function from X to Y and a is a domain morphism from A to B such that a simulates f on the level of realizers, i.e. $a(r(x)) = s(f(x))$ for any $x \in$

X, or more concisely : $a \circ r = s \circ f$. The category obtained in this way is known as the *Freyd cover* or the *sconing* of **Dom** , see e.g. [LS] and [FS].

Before giving the official definition of the category of intensional realizability sets we discuss to some more detail our requirements for the category **Dom** .The general framework will be a cartesian closed category $\mathcal{C}$ together with a terminal object 1 which is a generator, i.e. the functor $\mathcal{C}(1, \_) : \mathcal{C} \to \mathbf{Set}$ is faithful . Cartesian closed categories where the terminal object is a generator will be called *extensional* . This is a stronger notion than the notion of so called *concrete* (cartesian closed) categories which are (cartesian closed) categories $\mathcal{C}$ which come as equipped with a functor $U : \mathcal{C} \to \mathbf{Set}$ which is faithful and reflects identity morphisms. Of course, concrete cartesian closed categories need not have a terminal objects (as e.g. the category of infinite sets).
Beyond the category **Set** itself, typical and for our purposes more interesting examples of such extensional cartesian closed categories are the category of directedly complete partial orders - not necessarily with a bottom element - and Scott continous maps and its prominent full subcategories : SFP objects, consistently complete ω-algebraic cpos (the so called *Scott domains*) or the coherently complete ω-algebraic cpos. Especially interesting are the effective versions of these subcategories as discussed e.g. in [Plo].
Of course, one could equally well take one of the more recently introduced more general categories of effective domain such as the category of *complete extensional pers*, see [FMRS], or Martin Hyland's category of *Σ-replete pers*. Those appear as full subcategories of the category PERω (of partial equivalence relations on ω) and realizable morphisms between them, often also called the category of *modest sets*, which itself constitutes a very rich extensional cartesian closed category.
The choice of extensional cartesian closed category is motivated by their following property : *functions which are defined in a sufficiently constructive way appear as morphisms in the extensional cartesian closed category.*
Anyway the definitions and constructions given below will be independent from the specific choice of the underlying category **Dom** of domains and domain morphisms. But whenever we need special properties of **Dom** we will explicitely state them !

2.3 The Category IRS of Intensional Realizability Sets

Now we shall give a precise definition of the category IRS of intensional realizability sets, introduce some terminology for further reference and study some of its categorical properties. In the sequel we shall use π_0 and π_1 to denote the projections on the first and second component, respectively, both in the category of sets and in the underlying category of domains.

Definition 1.1

The category IRS of *intensional realizability sets* and *intensional realizable maps* is given by the following data.

Objects of IRS are triples $\underline{X} = (X , A , r)$ where X is a set , A is a domain and $r : X \to A$ is a set-theoretic function associates with any object $x \in X$ its realizer $r(x) \in A$.
Objects of IRS are also called *intensional realizability sets* or *irs-s.*
For an intensional realizability set $\underline{X} = (X , A , r)$ we call X the *underlying set of* $\underline{X}$ which will be denoted by $\mathrm{Set}(\underline{X})$ and we call A the *underlying domain of potential realizers of* $\underline{X}$ which will be denoted by $\mathrm{Dom}(\underline{X})$ and we call r the *underlying realizability function of* $\underline{X}$ which will be denoted by $r(\underline{X})$.

For intensional realizability sets $X_1 = (X_1 , A_1 , r_1)$ and $X_2 = (X_2 , A_2 , r_2)$ the morphisms from X_1 to X_2 are the pairs $a = (f , a)$ where $f : X_1 \to X_2$ is an arbitrary set-theoretic function and $a : A_1 \to A_2$ is a domain morphisms such that $r_2 \circ f = a \circ r_1$.
Morphisms $\alpha = (f , a)$ and $\beta = (g , b)$ are equal as morphisms from X_1 to X_2 in IRS iff f and g are equal as set-theoretic functions and a and b are equal as domain morphisms.
Morphisms in the category IRS will also be called *intensional realizable maps.*
For an intensional realizable morphism $\alpha = (f , a)$ we call f the underlying function of the morphism a which will be denoted by $\mathrm{fun}(\alpha)$ and a the underlying realizer of a which will be denoted by $\mathrm{real}(\alpha)$.

Composition of morphisms in IRS is defined by composition of the components, i.e.

$$(g , b) \circ (f , a) := (g \circ f , b \circ a)$$

For an intensional realizability set $\underline{X} = (X , A , r)$ the identity morphism $\mathrm{id}_{\underline{X}}$ is given by the pair $(\mathrm{id}_X , \mathrm{id}_A)$.
It is straightforward to check that these data define a category. ◊

Remark. The *extensional quotient* of this category is obtained by identifying those morphism which differ at most in the second component.

Now we shall prove some important categorical properties of IRS . For this purpose we first need the definition of *display map* which, intuitively speaking, is that kind morphism in IRS which represents or *displays* a *families of types.*

Definition 1.2

A morphism $\alpha : \underline{Y} \to \underline{X}$ in IRS is a *display map* iff $\mathrm{real}(\alpha) : \mathrm{Dom}(\underline{Y}) \to \mathrm{Dom}(\underline{X})$ is a projection map, i.e. there is a domain A such that $\pi_0 : \mathrm{Dom}(\underline{X}) \times A \to \mathrm{Dom}(\underline{X})$ is isomorphic to $\mathrm{real}(\alpha) : \mathrm{Dom}(\underline{Y}) \to \mathrm{Dom}(\underline{X})$ in $\mathbf{Dom} / \mathrm{Dom}(\underline{X})$. ◊

Remark. Intuitively, display maps with codomain $\underline{X}$ correspond to those families of intensional realizability sets where *all elements of the family have the same underlying domain of potential realizers*.
A slight generalization of the definition of display map can be obtained by relaxing the condition that $\mathrm{real}(\alpha)$ is a projection map to the more general condition that $\mathrm{real}(\alpha)$ is a continuous fibration in the sense of [PSH].

Theorem 2.3

In the category IRS display maps are stable under pullbacks along arbitary morphisms and whenever $\alpha : \underline{Y} \to \underline{X}$ and $\beta : \underline{Z} \to \underline{Y}$ are display maps then the functor

$$\mathrm{IRS} / \underline{Y} (\alpha^*\_ , \beta) : (\mathrm{IRS} / \underline{X})^{\mathrm{op}} \to \mathbf{Set}$$

is representable by a display map $\gamma : \underline{P} \to \underline{X}$.

Proof : The proof is essentially the same as the proof in the next section that **ICont** is a contextual category with products of families of types. ◊

The only obstacle preventing IRS from being locally cartesian closed is that in general IRS does not have equalizers. But if the underlying category **Dom** itself has equalizers - as it is the case for cpos (not necessarily having a bottom element) and continuous maps, complete Σ-extensional pers and Σ-replete pers - then the category IRS is guaranteed to have all equalizers and, furthermore, to be locally cartesian closed. This will be shown in the next theorem.

Theorem 2.4

If the underlying category **Dom** of domains has equalizers then the category IRS itself is locally cartesian closed.

Proof : It is a well known fact that the Freyd cover of a category having all finite limits has also all finite limits.
Thus it remains to show is that the category IRS is locally cartesian closed.
Let $\alpha : \underline{Y} \to \underline{X}$ and $\beta : \underline{Z} \to \underline{Y}$ be morphisms in IRS.
For reasons of notational convenience we will write X, Y and Z for $\mathrm{Set}(\underline{X})$, $\mathrm{Set}(\underline{Y})$ and $\mathrm{Set}(\underline{Z})$, respectively, and A, B and C for $\mathrm{Dom}(\underline{X})$, $\mathrm{Dom}(\underline{Y})$ and $\mathrm{Dom}(\underline{Z})$, respectively, and r, s and t for $r(\underline{X})$, $r(\underline{Y})$ and $r(\underline{Z})$, respectively. Furthermore for the same reason we will write $f : Y \to X$ and $g : Z \to Y$ for $\mathrm{fun}(\alpha)$ and $\mathrm{fun}(\beta)$, respectively, and $a : B \to A$

and $b : C \rightarrow B$ for $\mathrm{real}(\alpha)$ and $\mathrm{real}(\beta)$, respectively.

The morphism $\Pi_\alpha\beta : \underline{P} \rightarrow \underline{X}$ is described as follows. The underlying set $\mathrm{Set}(\underline{P})$ - which we temporarily call P - is defined as the set of all triples (x , h , c) such that

$x \in X$

$h : f^{-1}(\{x\}) \rightarrow Z$ with $g(h(y)) = y$ for all y with $f(y) = x$

$c : B \rightarrow C$ is a domain morphism satisfying
$t(h(y)) = c(s(y))$ for all y with $f(y) = x$

The underlying domain $\mathrm{Dom}(\underline{P})$ is defined as $A \times [B \rightarrow C]$ and the underlying realizability map $r(\underline{P}) : P \rightarrow A \times [B \rightarrow C]$ is defined as

$r(\underline{P})(x , h , c) = (r(x) , c)$ for all $(x , h , c) \in P$

Next we describe the morphism $\Pi_\alpha\beta$ which we temporarily call γ:

for all $(x , h , c) \in P$:

$\mathrm{fun}(\gamma)(x , h , c) = x$

for all $(d , c) \in A \times [B \rightarrow C]$:

$\mathrm{real}(\gamma)(d , c) = d$

Finally, the evaluation morphism is described as follows.
Objects $(x , h , c) \in P$ and $y \in Y$ with $f(y) = x$ are mapped by the underlying function of the evaluation morphism to $h(y)$. The realizer for the evaluation morphism is the domain morphism mapping $(d , c) \in A \times [B \rightarrow C]$ and $b \in B$ to $c(b)$. ◊

Based on these categorical properties we shall define in the next section a contextual category with products of families of types which will serve as a (n extensional) model for the Logical Framework containing enough structure to interpret ICST.

2.4 An Intensional Contextual Category with Dependent Products

We shall define a contextual category **ICont** based on the category IRS of intensional realizability sets. This means that there is a full and faithful functor DEN forgetting **ICont** to

IRS . For a detailed explanation of the notion of contextual categoy see e.g. [Str1].

We inductively define a class ICONT of *contexts* together with an interpretation function DEN : ICONT $\rightarrow$ Ob(IRS) associating with every context an intensional realizability set and a mapping length : ICONT $\rightarrow \omega$ associating with every context its length.

(1) The *empty context* ε is an element of ICONT and length$[\varepsilon] = 0$.

$$\mathrm{DEN}[\varepsilon] = (\{*\} , 1 , r_1)$$

where $\{*\}$ is the underlying set of ε ,
1 is the terminal object in the category of domains
and r_1 is the unique function from $\{*\}$ to 1 .

(2) if $\Gamma \in$ ICONT with length$[\Gamma] = n$ and $\mathrm{DEN}[\Gamma] = (X , A , r)$
and B is a domain and $F : X \rightarrow \mathrm{Ob(IRS)}$
such that for all $x \in X$ the underlying domain of potential realizers of F(x) is B ,
i.e. $\mathrm{Dom}(F(x)) = B$ for all $x \in X$,
then

$(\Gamma, B , F) \in$ ICONT

$\text{length}\,[(\Gamma, B , F)] = n + 1$

$\mathrm{DEN}\,[(\Gamma, B , F)] = (S , A \times B , r')$ where

$S = \{ (x , y) \mid x \in X \text{ and } y \in \mathrm{Set}(F(x)) \}$ and

$r' (x , y) = (r(x) , \mathrm{real}(F(x))(y))$ for all $x \in X$ and $y \in \mathrm{Set}(F(x))$

The objects of the contextual category **ICont** are the elements of ICONT which are often simply called *contexts*.

The morphisms from context Γ to context Δ are the morphisms in IRS from DEN[Γ] to DEN[Δ] and we simply write $\alpha : \Gamma \rightarrow \Delta$ instead of $\alpha : \mathrm{DEN}[\Gamma] \rightarrow \mathrm{DEN}[\Delta]$.
Composition of morphisms and identity morphisms are inherited from IRS .
A context Γ is the *father of* the context Δ iff Δ is of the form (Γ, B , F) .

We write $\Gamma \lhd \Delta$ to indicate that context Γ is the father of the context Δ .

Of course, the relation $\lhd$ induces a tree structure on ICONT where ε is the root of the tree and a terminal object in **ICont** .

If $\Gamma \lhd \Delta$ then the *canonical projection* $p(\Delta) : \Delta \to \Gamma$ is defined as follows.
Let $\mathrm{DEN}[\Gamma] = (X , A , r)$. As $\Gamma \lhd \Delta$ we know that $\Delta = (\Gamma, B , F)$ for some domain B and some $F : X \to \mathrm{Ob(IRS)}$ such that $\mathrm{Dom}(F(x)) = B$ for all $x \in X$. Then $p(\Delta)$ is defined as the morphism whose components

$$\mathrm{real}(p(\Delta)) : A \times B \to A$$

$$\mathrm{fun}(p(\Delta)) : \mathrm{Set}(\mathrm{DEN}[\Delta]) \to \mathrm{Set}(\mathrm{DEN}[\Gamma])$$

are simply the projections on the first components, i.e.

$$\mathrm{fun}(p(\Delta))(x,y) = x \qquad \text{for all } x \in X ,\ y \in \mathrm{Set}(F(x))$$

$$\mathrm{real}(p(\Delta))(a,b) = a \qquad \text{for all } a \in A ,\ b \in B$$

We employ the notation $\alpha : \Delta \longrightarrow\!\!\rhd\, \Gamma$ in order to indicate that $\alpha : \Delta \to \Gamma$ is a canonical projection, i.e. $\alpha = p(\Delta)$.

For any context morphism $\alpha : \Theta \to \Gamma$ and any context $\Delta = (\Gamma, B , F)$ there is a distinguished context

$$\alpha^*\Delta := (\Theta , B , F \circ \mathrm{fun}(\alpha))$$

and a distinguished context morphism

$$q(\alpha, \Delta) : \alpha^*\Delta \to \Delta$$

with components

$$\mathrm{fun}(q(\alpha, \Delta)) : \mathrm{Set}(\alpha^*\Delta) \to \mathrm{Set}(\Delta)$$

$$\mathrm{real}(q(\alpha, \Delta)) : \mathrm{Dom}(\alpha^*\Delta) \to \mathrm{Dom}(\Delta)$$

which are defined as follows

$$\mathrm{fun}(q(\alpha, \Delta))(z,y) = (\mathrm{fun}(\alpha)(z)), y)$$

$$\text{for all } z \in \mathrm{Set}(\mathrm{DEN}(\Theta)) ,\ y \in \mathrm{Set}(F(\mathrm{fun}(\alpha)(z)))$$

$\text{real}(q(\alpha, \Delta))(c,b) = (\text{real}(\alpha)(c)), b)$

for all $c \in \text{Dom}(\text{DEN}(\Theta))$, $b \in B$

Next we show that these data actually define a contextual category, cf. [Str1].

First we show that for any context morphism $\alpha : \Gamma 1 \rightarrow \Gamma$ and any context $\Delta = (\Gamma, B, F)$ the morphisms $p(\alpha^*\Delta)$ and $q(\alpha, \Delta)$ constitute a pullback cone.

Let Θ be a context and $\beta : \Theta \rightarrow \Gamma 1$ and $\gamma : \Theta \rightarrow \Delta$ be morphisms with $\alpha \circ \beta = p(\Delta) \circ \gamma$. Then there exists a unique morphism $\mu : \Theta \rightarrow \alpha^*\Delta$ such that

$p(\alpha^*\Delta) \circ \mu = \beta$ and $q(\alpha, \Delta) \circ \mu = \gamma$

where

$\text{fun}(\mu)(z) = (\text{fun}(\beta)(z), \pi_1(\text{fun}(\gamma)(z)))$

for all $z \in \text{Set}(\text{DEN}(\Theta))$

and

$\text{real}(\mu)(c) = (\text{real}(\beta)(c), \pi_1(\text{real}(\gamma)(c)))$

for all $c \in \text{Dom}(\text{DEN}(\Theta))$

Next we shall define some further ingredients needed in order to have dependent products of families of types.
Let Γ, Δ, Θ be contexts with $\Gamma \lhd \Delta \lhd \Theta$. Then there exist domains A and B and functions $F : \text{DEN}[\Gamma] \rightarrow \text{Ob(IRS)}$, $G : \text{DEN}[\Delta] \rightarrow \text{Ob(IRS)}$ such that

$\Delta = (\Gamma, A, F)$ and $\Theta = (\Delta, B, G)$
Obviously, by definition of what a context is, for F and G it must hold that $\text{Dom}(F(x)) = A$ for all $x \in \text{Set}(\text{DEN}[\Gamma])$ and $\text{Dom}(G(z)) = B$ for all $z \in \text{Set}(\text{DEN}[\Delta])$.

Then there is a distinguished context

$$\Pi(\Theta) = (\Gamma, [A \to B], P)$$

where $P : \mathrm{Set}(\mathrm{DEN}[\Gamma]) \to \mathrm{Ob}(\mathrm{IRS})$ is defined as follows. For any $x \in \mathrm{Set}(\mathrm{DEN}[\Gamma])$ let $\mathrm{Dom}(P(x)) := [A \to B]$ and $\mathrm{Set}(P(x))$ be the set of all pairs (s, a) where $a : A \to B$ is a domain morphism and

$$s : \mathrm{Set}(F(x)) \to \bigcup \{ \mathrm{Set}(G(x,y)) \mid y \in \mathrm{Set}(F(x)) \}$$

such that for all $y \in \mathrm{Set}(F(x))$:

$$s(y) \in \mathrm{Set}(G(x,y)) \qquad \text{and}$$

$$a(\mathrm{real}(F(x))(y)) = \mathrm{real}(G(x,y))(s(y))$$

and the realizability function $r(P(x))$ is the projection on the second component, i.e.

$$r(P(x))(s, a) = a \qquad \text{for all } (s, a) \in \mathrm{Set}(P(x))$$

Furthermore there is a distinguished morphism

$$\mathrm{eval}(\Theta) : p(\Delta)^*\Pi(\Theta) \to \Theta \qquad \text{over } \Delta$$

which means that it satisfies the condition

$$p(\Theta) \circ \mathrm{eval}(\Theta) = p(p(\Delta)^*\Pi(\Theta))$$

where the components of $\mathrm{eval}(\Theta)$ are defined as follows :

$$\mathrm{fun}(\mathrm{eval}(\Theta))((x, y), (a, s)) = ((x, y), s(y))$$

for all $x \in \mathrm{Set}(\mathrm{DEN}[\Gamma])$, $y \in \mathrm{Set}(F(x))$, $(a, s) \in \mathrm{Set}(P(x))$

$$\mathrm{real}(\mathrm{eval}(\Theta))((c, d), a) = ((c, d), a(d))$$

for all $c \in \mathrm{Dom}(\mathrm{DEN}[\Gamma])$, $d \in A$, $a \in [A \to B]$

Now we show that these additional ingredients satisfy the conditions required for dependent product types.

Whenever τ is a *section* of Θ, i.e. $\tau : \Delta \to \Theta$ and $p(\Theta) \circ \tau = \mathrm{id}_\Delta$, then there is a *unique* morphism $\sigma : \Gamma \to \Pi(\Theta)$ which is a section of $\Pi(\Theta)$ and furthermore satisfies the condition

$$\mathrm{eval}(\Theta) \circ p(\Delta)^*\sigma = \tau$$

We first describe a section σ satisfying the characterizing equation and then prove its uniqueness.
The morphism σ is given by its components

$$\mathrm{fun}(\sigma) : \mathrm{Set}(\mathrm{DEN}[\Gamma]) \to \mathrm{Set}(\mathrm{DEN}[\Pi(\Theta)])$$

$$\mathrm{real}(\sigma) : \mathrm{Dom}(\mathrm{DEN}[\Gamma]) \to \mathrm{Dom}(\mathrm{DEN}[\Gamma]) \times [A \to B]$$

which are defined as follows. Let $a : \mathrm{Dom}(\mathrm{DEN}[\Gamma]) \to [A \to B]$ be the unique domain morphism such that for all $c \in \mathrm{Dom}(\mathrm{DEN}[\Gamma])$ and $d \in A$

$$\mathrm{real}(\tau)(c, d) = (c, d, a(c)(d))$$

For arbitrary $c \in \mathrm{Dom}(\mathrm{DEN}[\Gamma])$ we put

$$\mathrm{real}(\sigma)(c) = (c, a(c))$$

and for arbitrary $x \in \mathrm{Set}(\mathrm{DEN}[\Gamma])$ we put

$$\mathrm{fun}(\sigma)(x) = (x, (s, a(r(\mathrm{DEN}[\Gamma])(x))))$$

where $s : \mathrm{Set}(F(x)) \to \bigcup \{ \mathrm{Set}(G((x, y))) \mid y \in \mathrm{Set}(F(x)) \}$
is the unique function such that for all $y \in \mathrm{Set}(F(x))$

$$\mathrm{fun}(\tau)(x, y) = ((x, y), s(y))$$

It is straightforward, but tedious to check that σ satisfies the equation $\mathrm{eval}(\Theta) \circ p(\Delta)^*\sigma = \tau$.
For showing uniqueness suppose that σ' is an arbitrary section of $\Pi(\Theta)$ satisfying the condition $\mathrm{eval}(\Theta) \circ p(\Delta)^*\sigma' = \tau$.
We first show that $\mathrm{real}(\sigma') = \mathrm{real}(\sigma)$. As both σ' and σ are sections of $\Pi(\Theta)$ there are domain morphisms $a', a : \mathrm{Dom}(\mathrm{DEN}[\Gamma]) \to [A \to B]$ such that for all $c \in \mathrm{Dom}(\mathrm{DEN}[\Gamma])$

$$\mathrm{real}(\sigma')(c) = (c, a'(c)) \quad \text{and} \quad \mathrm{real}(\sigma)(c) = (c, a(c))$$

and therefore it is sufficient to show that $a' = a$.

As $\mathrm{eval}(\Theta) \circ p(\Delta)^*\sigma' = \tau$ we also have

$$\mathrm{real}(\mathrm{eval}(\Theta)) \circ \mathrm{real}(p(\Delta)^*\sigma') = \mathrm{real}(\tau)$$

That means that for all $c \in \mathrm{Dom}(\mathrm{DEN}[\Gamma])$ and $d \in A$:

$$\mathrm{real}(\mathrm{eval}(\Theta))(\mathrm{real}(p(\Delta)^*\sigma')(c, d)) = \mathrm{real}(\tau)(c, d)$$

and as we know that $\mathrm{real}(p(\Delta)^*\sigma')(c, d) = ((c, d), a'(c))$ we get

$$\mathrm{real}(\mathrm{eval}(\Theta))((c, d), a'(c)) = \mathrm{real}(\tau)(c, d)$$

and as we know that $\mathrm{real}(\mathrm{eval}(\Theta))((c, d), a'(c)) = (c, a'(c)(d))$ we get

$$(c, a'(c)(d)) = \mathrm{real}(\tau)(c, d)$$

Thus we have shown that for all $c \in \mathrm{Dom}(\mathrm{DEN}[\Gamma])$ and $d \in A$:

$$\mathrm{real}(\tau)(c, d) = (c, a'(c)(d))$$

But from the definition of σ we know that for all $c \in \mathrm{Dom}(\mathrm{DEN}[\Gamma])$ and $d \in A$:

$$\mathrm{real}(\tau)(c, d) = (c, a(c)(d))$$

Thus by transitivity of equality we have that for all $c \in \mathrm{Dom}(\mathrm{DEN}[\Gamma])$ and $d \in A$:

$$(c, a'(c)(d)) = (c, a(c)(d))$$

But as the category of domains is a model for λ-calculus with η-rule this implies that

$$a' = a$$

Analogously, one can prove that $\mathrm{fun}(\sigma') = \mathrm{fun}(\sigma)$.

After having found a model of the Logical Framework we shall turn to the task of interpreting ICST in it.

2.5 A Model of Intensional Constructive Set Theory in **ICont**

In order to interpret ICST in **ICont** we first have to define an object Set and an object El with Set $\lhd$ El .
The idea for interpreting Set is that it should be a collection of small types. Now, what is a good criterion of smallness for intensional realizability sets ? In the case of the usual PERω model in **ω-Set** the criterion was that *every object is determined uniquely by its realizer.* If we transfer this idea to intensional relizability sets then the condition of smallness for (X, A, r) is that for all $x, x' \in X$ if $r(x) = r(x')$ then $x = x'$, in other words, that r is injective and therefore essentially X is a subset of A . But we have to be careful as in the case of **ω-Set** we have only one collection of realizers, namely ω . Now in the case of intensional realizability sets any domain serves as a collection of realizers. For small types the underlying domain of potential realizers should be a domain of algorithmic objects fixed in advance.
For the purposes of interpreting intensional constructive set theory we assume that this fixed domain of algorithmic objects - which we call from now on D - satisfies the following domain equation

$$D \;\cong\; [D \rightarrow D]_\perp \uplus [D \times D]_\perp \uplus [D + D] \uplus D_\perp \uplus \{0\}_\perp \uplus D_\perp$$

Here $\uplus$ denotes the *coalesced sum* of two domains where the bottom elements of each sum are identified and $+$ denotes the *separated sum* where a new bottom element is added to the disjoint union.
Obviously, this domain equation is the one suitable for interpreting the underlying functional language of intensional constructive set theory when lazy evaluation is assumed (see e.g. [ML4]).

Now we will give the inetrpretation of ICST in **ICont**.

2.5.1 Interpretation of Set and El

The interpretation of Set is (ε, F) with $F(*) = (\mathcal{P}(D), 1, r)$ where r is the unique function from $\mathcal{P}(D)$ to $\{*\}$, the underlying set of 1 .

The object El with Set $\lhd$ El is (Set, D, G) with $G(*, A) = (A, D, \text{in}_A)$ for any $A \in \mathcal{P}(D)$ where $\text{in}_A : A \rightarrow D$ denotes the embedding of A into D .

In the sequel quite sloppily we do not distinguish between Set and $F(*)$, El and G , and instead of $G(*, A)$ we simply write $\text{El}(A)$ or even El A .

Similarly, when (s, a) is a morphism from ε to some Γ of length 1 then we always omit the trivial argument $*$, i.e. we write s for $s(*)$ and a for $a(*)$.

2.5.2 Interpretation of Π, fun and F

If $A \in \mathrm{Set}$ and $B \in \{x : \mathrm{El}\ A\}\ \mathrm{Set}$ then

$$\Pi\, A\, B = \{\ \mathrm{map}(h) \mid h \in [D \to D] \text{ and } h(d) \in B(d) \text{ for all } d \in A\ \}$$

where $\mathrm{map} : [D \to D] \to D$ is the embedding into D of the first summand $[D \to D]$ of the right hand side of the domain equation characterising the algorithmic universe D.

If $(s, h) \in \{x : \mathrm{El}\ A\}\ \mathrm{El}\ (B\ x)$ then

$$\mathrm{fun}\ A\ B\ (s, h) = \mathrm{map}(h)$$

and fun is realized by the domain morphism $m : (1 \times [1 \to 1]) \times [D \to D] \to D$ described by the equation $m((*, \mathrm{id}_1), h) = \mathrm{map}(h)$ for all $h \in [D \to D]$.

Next we will describe the elimination operator F.
If $C \in \{p : \mathrm{El}\ (\Pi\ A\ B)\}\ \mathrm{Set}$ and $d \in \{f : \{x : \mathrm{El}\ A\}\ \mathrm{El}\ (B\ x)\}\ \mathrm{El}\ (C\ (\mathrm{fun}\ A\ B\ f))$ then the interpretation of $F\ A\ B\ C\ d$ is defined as follows. If $c \in \mathrm{El}\ (\Pi\ A\ B)$ then $c = \mathrm{map}(h)$ for some $h \in [D \to D]$ such that $(s, h) \in \{x : \mathrm{El}\ A\}\ \mathrm{El}\ (B\ x)$ for a unique s which in fact is the restriction of h to A. Of course, c is realized by itself. Then we put

$$\mathrm{fun}(F\ A\ B\ C\ d)\ c = \mathrm{fun}(d)\ (s, h)$$

and

$$\mathrm{real}(F\ A\ B\ C\ d)\ c = \mathrm{real}(d)\ h \qquad \text{if } c = \mathrm{map}(h) \text{ for some } h \in [D \to D]$$

$$\mathrm{real}(F\ A\ B\ C\ d)\ c = \bot_D \qquad \text{otherwise}$$

It is obvious from this definition of F that in our model the equation

$$F\ A\ B\ C\ d\ (\mathrm{fun}\ A\ B\ f) = d\ f$$

is valid for all admissible objects A, B, C, d, f.

Obviously, this is not the only way to define real(F A B C d) as there is a lot of other different domain morphisms $m : D \to D$ satisfying the condition that for all $h \in [D \to D]$

$$m\,(\text{map}(h)) \;=\; \text{real}(d)\; h \;=\; \text{fun}(F\ A\ B\ C\ d)\ (\text{map}(h))$$

For example for any arbitrary $d_0 \in D$ we can put $m(c) = d_0$ for all $c \in D$ with $c \neq \perp_D$ and $c \neq \text{map}(h)$ for all $h \in [D \to D]$.
Therefore uniqueness of elimination fails in a very strongly as if $e \in \{c : \text{El}\ (\Pi\ A\ B)\}\ \text{El}(C\ c)$ with $\text{real}(e)(c) = d_0$ for all $c \in D$ with $c \neq \perp_D$ and $c \neq \text{map}(h)$ for all $h \in [D \to D]$ then in our model it holds that e is different from

$$e' := F\ A\ B\ C\ ([f : \{x : \text{El}\ A\}\ \text{El}\ (B\ x)]\ e\ (\text{fun}\ A\ B\ f))$$

as for all $c \in D$ with $c \neq \perp_D$ and $c \neq \text{map}(h)$ for all $h \in [D \to D]$

$$\text{real}(e)(c) \;=\; \perp_D \quad \text{whereas} \quad \text{real}(e')(c) \;=\; d_0$$

and for almost all choices d_0 is different from $\perp_D$.

2.5.3 Interpretation of Σ, pair and E

If $A \in \text{Set}$ and $B \in \{a : \text{El}\ A\}\ \text{Set}$ then $\Sigma\ A\ B$ is the subset of D consisting of all objects $\text{pr}(d1,d2) \in D$ such that $d1 \in \text{El}\ A$ and $d2 \in \text{El}(B\ d1)$.
Of course, the mapping pr is the embedding into D of the second summand $D \times D$ of the right hand side of the domain equation characterising the algorithmic universe D.

The underlying function of the morphism pair maps any $d1 \in \text{El}\ A$ and $d2 \in \text{El}(B\ d1)$ to the object $\text{pr}(d1,d2)$ in $\Sigma\ A\ B$.
It is realized by the domain morphism mapping any $d1, d2 \in D$ to $\text{pr}(d1,d2) \in D$.

The elimination operation E is defined as follows.
If $A \in \text{Set}$, $B \in \{a : \text{El}\ A\}\ \text{Set}$ and $d \in \{a : \text{El}\ A\}\{b : \text{El}\ (B\ a)\}\ \text{El}(C(\text{pair}(A,B,a,b)))$ then the denotation of $E\ A\ B\ C\ d$ is described as follows. Its underlying function maps any object of the form $\text{pr}(d1,d2) \in \Sigma\ A\ B$ to $d\ d1\ d2$. If f is any realizer for d then the realizer for the object $E\ A\ B\ C\ d$ is the domain morphism mapping any argument of the form $\text{pr}(d1,d2)$ to the object $f\ d1\ d2$ and any other argument to $\perp_D$.
Obviously, E satisfies the corresponding β-like rule.
But uniqueness of elimination does not hold because instead of $E\ A\ B\ C\ d$ we can take a morphism with the same underlying function as $E\ A\ B\ C\ d$ but with a realizer which on arguments of the form $\text{pr}(d1,d2)$ behaves like the realizer of $E\ A\ B\ C\ d$ but gives $\perp_D$ as result if the argument is $\perp_D$ and some arbitrary fixed object $d_0 \in D$ in all other cases.

2.5.4 Interpretation of Id , r and J and K

If $A \in Set$ and $a , b \in El\ A$ then the set $Id\ A\ a\ b = \{ id(d) \mid a = d = b \}$ where $id : D \rightarrow D$ is the embedding into D of the 4th component of the right hand side of the characterizing domain equation for D .
Of course, $Id\ A\ a\ b$ is the empty set if $a \neq b$ and otherwise $Id\ A\ a\ a = \{ id(a) \}$.

The mapping r associates with any set A and any $a \in A$ the object $id(a) \in Id\ A\ a\ a$.
The morphism r is realized by the domain morphism which for any $d \in D$ gives the element $id(d) \in D$.

Next we shall describe the interpretation of the elimination operation J .
Let $A \in Set$, $C \in \{a : El\ A\}\{b : El\ A\}\{c : El\ (Id\ A\ a\ b)\}\ Set$ and $d \in \{a : El\ A\}\ El\ (C\ a\ a\ (r\ A\ a))$. Then there is only one possible choice for the underlying set theoretic function of the object $J\ A\ C\ d$: if $a , b \in A$ and $c \in El\ (Id\ A\ a\ b)$ then $c = id(a)$ and $a = b$ and therefore it must hold that

$$fun(J\ A\ C\ d)\ a\ b\ c \ = \ fun(d)\ (a)$$

as required by the conversion rule for the eliminator J .

But there are a lot of different choices of how to define $real(J\ A\ C\ d)$. Our canonical choice is the following :

$$real(J\ A\ C\ d)\ a\ b\ id(x) \ = \ real(d)\ x$$

$$real(J\ A\ C\ d)\ a\ b\ c \ = \ \perp_D \qquad \text{if } c \neq id(x) \text{ for all } x \in D$$

Under this interpretation the β-like rule for J is satisfied but not the η-like rule.
The reason is that for any given $d \in \{a : El\ A\}\ El\ (C\ a\ a\ (r\ A\ a))$ there are several mappings $e \in \{a : El\ A\}\{b : El\ A\}\{c : El\ (Id\ A\ a\ b)\}\ El\ (C\ a\ b\ c)$ which extend d but are different from $J\ A\ C\ d$.
Of course, the underlying function $fun(e)$ must always be equal to $fun(J\ A\ C\ d)$ in order to validate

$$J\ A\ C\ d\ a\ a\ (r\ A\ a) \ = \ d\ a$$

the conversion law for the eliminator J .

But for $fun(e)$ there is plenty of choice as shown by the following examples.

Let $d_0 \in D$ be arbitrary and define

$$\text{real}(e)\ a\ b\ \text{id}(x) = \text{real}(d)\ x = \text{real}(J\ A\ C\ d)\ a\ b\ \text{id}(x)$$

$$\text{real}(e)\ a\ b \perp_D = \perp_D = \text{real}(J\ A\ C\ d)\ a\ b \perp_D$$

but for arguments a, b, c with $c \neq \perp_D$ and $c \neq \text{id}(x)$ for all $x \in D$ we put

$$\text{real}(e)\ a\ b\ c = d_0$$

Then for any arbitrary choice of d_0 as long as $d_0 \neq \perp_D$ we have for all $c \in D$ with $c \neq \perp_D$ and $c \neq \text{id}(x)$ for all $x \in D$ that

$$\text{real}(e)\ a\ b\ c = d_0 \neq \perp_D = \text{real}(J\ A\ C\ d)\ a\ b\ c$$

There are two further examples illustrating that one may define real(e) in a way that it shows a behaviour different from real(J A C d) even for arguments a, b, c where $c = \text{id}(x)$ for some $x \in D$.
Either one puts for all $a, b, c \in D$

$$\text{real}(e)\ a\ b\ c = \text{real}(d)\ a$$

or one puts for all $a, b, c \in D$

$$\text{real}(e)\ a\ b\ c = \text{real}(d)\ b$$

Then for *both* definitions of e one has

$$\text{real}(e)\ a\ a\ \text{id}(a) = \text{real}(d)\ a = \text{real}(J\ A\ C\ d)\ a\ a\ \text{id}(a)$$

guaranteeing that *also on the level of realizers* the defining equation

$$J\ A\ C\ d\ a\ a\ (r\ A\ a) = d\ a$$

for J is valid in the model.

Although the model under consideration is very intensional in the sense that that it strongly refutes the η-like rule for J, it is nevertheless possible to interpret our new elimination operator K in a natural way which is very similar to the interpretation of J. This is a further indication of the naturality of our new eliminator K.

If A is a set, $C \in \{a : El\ A\}\ \{c : El\ (Id\ A\ a\ a)\}\ Set$ and $d \in \{a : El\ A\}\ El\ (C\ a\ (r\ A\ a))$ then $K\ A\ C\ d$ is defined as follows. For any $a \in A$ and $c \in El\ (Id\ A\ a\ a)$ it necessarily holds that $c = id(a)$ and therefore we have

$$fun(K\ A\ C\ d)\ a\ c \ = \ fun(d)\ (a)$$

A natural choice for defining the realizer part of $K\ A\ C\ d$ is the following

$$real(K\ A\ C\ d)\ a\ id(x) \ = \ real(d)\ x$$

$$real(K\ A\ C\ d)\ a\ c \ = \ \perp_D \qquad \text{if } c \neq id(x) \text{ for all } x \in D$$

which strongly resembles the interpretation of J.

Again, as in the case of J, under this interpretation the β-like rule for K is satisfied but not the η-like rule. The reason is that for any given $d \in \{a : El\ A\}\ El\ (C\ a\ (r\ A\ a))$ there are several mappings $e \in \{a : El\ A\}\{c : El\ (Id\ A\ a\ a)\}\ El\ (C\ a\ c)$ which extend d but are different from our chosen interpretation of $K\ A\ C\ d$.
Of course, the underlying function $fun(e)$ must always be equal to $fun(K\ A\ C\ d)$ in order to validate

$$K\ A\ C\ d\ a\ (r\ A\ a) \ = \ d\ a$$

the conversion law for our new eliminator K.

But, again, for $fun(e)$ there is plenty of choice as shown by the following examples.
Let $d_0 \in D$ be arbitrary and define

$$real(e)\ a\ id(x) \ = \ real(d)\ x \ = \ real(K\ A\ C\ d)\ a\ id(x)$$

$$real(e)\ a \perp_D \ = \ \perp_D \ = \ real(K\ A\ C\ d)\ a \perp_D$$

but for arguments a, c with $c \neq \perp_D$ and $c \neq id(x)$ for all $x \in D$ we put

$$real(e)\ a\ c \ = \ d_0$$

Then for any arbitrary choice of d_0 as long as $d_0 \neq \perp_D$ we have for all $c \in D$ with $c \neq \perp_D$ and $c \neq id(x)$ for all $x \in D$ that

$$real(e)\ a\ c \ = \ d_0 \ \neq \perp_D \ = \ real(K\ A\ C\ d)\ a\ c$$

And there is a further example illustrating that one may define real(e) in a way that it shows a behaviour different from real(K A C d) even for arguments a , c where c = id(x) for some x ∈ D .
Simply put for all a , b , c ∈ D

real(e) a c = real(d) a

and one gets that

real(e) a id(a) = real(d) a = real(K A C d) a id(a)

guaranteeing that *also on the level of realizers* the defining equation

K A C d a (r A a) = d a

for K is valid in the model.

2.5.5 Interpretation of N , 0 , succ and R and other inductive types

Finally we show how to interpret the inductive type N of natural numbers. The set N is the least subset of D containing 0 and closed under the operation incr . Here incr denotes the embedding into D of the last summand of the right hand side of the domain equation characterizing D .

The interpretation of zero is 0 and the interpretation of succ is incr .

Next we describe the interpretation of the primitive recursor R .
For any family A : N → Set , d ∈ A zero and e : {n:N} (A n) → A (succ n) we define the realizer part real(R A d e) of the object R A d e ∈ {n:N} A n as the *least* domain morphism f satisfying the following two equations

f 0 = d

f (incr x) = real(e) x (f x)

By its realizer part the object R A d e is determined uniquely as A is a family of sets and objects of sets are uniquely determined by their realizers.

One can see immediately that the domain morphism real(R A d e) gives $\perp_D$ as result whenever it is applied to an argument which is different from 0 and not of the form incr x for some $x \in D$.

There exist other objects $h \in \{n{:}N\}\ A\ n$ with real(h) different from real(R A d e) although fun(h) = fun(R A d e) which is necessary for making h satisfy the equations

$$h\ 0 = d$$

$$h\ (\mathrm{succ}\ n) = e\ n\ (h\ n)$$

typical for R A d e .
Such h can be obtained by taking for real(h) any *non-minimal* solution of the equations

$$f\ 0 = d$$

$$f\ (\mathrm{incr}\ x) = \mathrm{real}(e)\ x\ (f\ x)$$

as e.g. the least g satisfying the equations

$$g\ 0 = d$$

$$g\ (\mathrm{incr}\ x) = \mathrm{real}(e)\ x\ (g\ x)$$

$$g\ c = d_0 \qquad \text{whenever } c \neq 0 \text{ and } c \neq \mathrm{incr}\ x \text{ for all } x \in D$$

where d_0 is an arbitrary element of D different from $\perp_D$.

As both R A d e and such h satisfy the typical equations for R A d e and nevertheless are different this demonstrates that the eliminator R does not satisfy the η-like rule for R .

Of course, other inductive types can be interpreted in our model as well provided that the domain D is chosen appropriately. This can be achieved by adding for any constructor an appropriate summand to the right hand side of the domain equation characterizing D .
E. g. if one wants to interpret Martin-Löf's W-types (Wellfounded Trees) in order to interpret the constructor sup one needs a further summand of the form $(D \times [D \to D])_\perp$.

2.6 The Models Obtained as Freyd Covers of Categories of Domains Are Semi-Intensional But Not Fully Intensional

Next we will check which of our criteria for intensionality (cf. 2.1) are valid in this model and which are refuted.

Theorem 2.5

The sequent

$$A : \mathrm{Set}\,,\; x : A\,,\; y : A\,,\; z : \mathrm{Id}\; A\; x\; y \vdash x = y \in A$$

- expressing that propositional equality reflects to judgemental equality - is not valid.

Proof : Let Γ temporarily be a name for the context $A : \mathrm{Set}\,,\; x : A\,,\; y : A\,,\; z : \mathrm{Id}\; A\; x\; y$.
The realizer part of the interpretation of x relative to the context Γ is the domain morphism f characterized by the equation

$$f\; u\; d1\; d2\; d3 \;=\; d1$$

and the realizer part of the interpretation of y relative to the context Γ is the domain morphism g characterized by the equation

$$g\; u\; d1\; d2\; d3 \;=\; d2$$

As f and g are different domain morphisms the interpretations of x and y relative to the context Γ are different. $\Diamond$

There is an important strengthening of the previous theorem expressing that for any set A and any object $a \in A$ the objects $z \in \mathrm{Id}\; A\; a\; a$ are not uniformly judgementally equal to the canonical object $r\; A\; a \in \mathrm{Id}\; A\; a\; a$.

Theorem 2.6

For *any* $A \in \mathrm{Set}$ and *any* $a \in A$ the following sequent is not valid

$$z : \mathrm{Id}\; A\; a\; a\,,\; p : \mathrm{Id}\; (\mathrm{Id}\; A\; a\; a)\; (r\; A\; a)\; z \vdash r\; A\; a = z \in \mathrm{Id}\; A\; a\; a$$

Proof : Let Γ temporarily be a name for the context $z : \mathrm{Id}\; A\; a\; a\,,\; p : \mathrm{Id}\; (\mathrm{Id}\; A\; a\; a)\; (r\; A\; a)\; z$.
The realizer part of the interpretation of $r\; A\; a$ relative to the context Γ is the domain morphism f satisfying the following characterizing equation

$$f\ d1\ d2 = id(a)$$

and the realizer part of the interpretation of z relative to the context Γ is the domain morphism g satisfying the following characterizing equation

$$g\ d1\ d2 = d1$$

This shows that the interpretations of $r\ A\ a$ and z relative to the context Γ are different in their realizer parts. ◊

Theorem 2.7

The identity reflection principle is also valid in our model : for any $A \in Set$ and $t, s \in A$ if there is an object $p \in Id\ A\ t\ s$ then t and s are equal objects of type A .

Proof : If $p \in Id\ A\ t\ s$ then the type $Id\ A\ t\ s$ is non-empty and t and s must be equal as otherwise the type $Id\ A\ t\ s$ would be empty due to our definition (c.f. 2.5.4). ◊

But the models obtained as Freyd covers of some category of domains are *semi-intensional* in the sense that it holds *w.r.t. arbitrary contexts* that if t and s are propositionally equal objects of set A then for any family of sets B indexed over A it holds that $B(s)$ and $B(t)$ are judgementally equal as sets.

Theorem 2.8

The sequent

$$A : Set\ ,\ B : A \rightarrow Set\ ,\ x : A\ ,\ y : A\ ,\ z : Id\ A\ x\ y \vdash B(x) = B(y) \in Set$$

is valid.

Proof : According to our definition (cf. 2.5.1) the type Set carries the trivial realizability structure, i.e. the domain of potential realizers of Set is 1 .
Therefore even relative to the context $A : Set , B : A \rightarrow Set , x : A , y : A , z : Id\ A\ x\ y$ the interpretations of $B(x)$ and $B(y)$ have the same realizer part.
But also the function part of their interpretation is equal as whenever $z \in Id\ A\ x\ y$ then according to the definition of Id the objects x and y must be equal - as otherwise $Id\ A\ x\ y$ were empty - and therefore also $B(x)$ and $B(y)$ are equal. ◊

Expectedly also the extensionality principle for functions is wrong in our model(s) as shown by the following theorem.

Theorem 2.9

The sequent

$$A : \mathrm{Set}\, , B : A \to \mathrm{Set}\, , f , g : \Pi\, A\, B\, , p : \{x{:}A\}\ \mathrm{Id}\ (B\ x)\ (f\ x)\ (g\ x) \vdash \mathrm{Id}\ (\Pi\, A\, B)\ f\ g$$

is not valid.

Proof : This sequent is not valid already for very simple instances of A and B, e.g. if one choses for A the set $\{0\}$ and for B the constant family with value A then there are (many examples of) two different functions f and g both mapping 0 to 0 but behaving differently at arguments different form 0. But for all such f and g the type $\{x{:}A\}$ Id (B x) (f x) (g x) is inhabited although the set Id (Π A B) f g is empty. $\Diamond$

As a further illustration of the power of our models we show that two very simple terms of type $N \to N$ have different denotations in our models although one can formally prove their pointwise propositional equality.
Actually it are the two terms of type $N \to N$ which have been introduced at the beginning of Chapter 1 to illustrate the difference between extensional and intensional equality of functions.

Theorem 2.10

In all models constructed as Freyd covers of some category of domains the following two terms have a different denotation

$$t \equiv [x : N]\ x \qquad \text{and} \qquad s \equiv [x : N]\ R\ ([x{:}N]\ N)\ 0\ ([x : N][y : N]\ \mathrm{succ}\ y)$$

although their pointwise propositional equality can be derived formally.

Proof : Obviously, we have real(t) $= \mathrm{id}_D$ but real(s) is a domain morphism which according to our interpretation of R (cf. 2.5.5) maps any $d \in D$ with $d \neq 0$ and $d \neq \mathrm{incr}(x)$ for all $x \in D$ to $\perp_D$. Therefore we have e.g.

$$\mathrm{real}(t)(\mathrm{map}(\mathrm{id}_D)) = \mathrm{map}(\mathrm{id}_D) \neq \perp_D = \mathrm{real}(s)(\mathrm{map}(\mathrm{id}_D))$$

showing that real(t) $\neq$ real(s) . $\Diamond$

Of course, this last result could also serve as a (less economical) proof of Theorem 2.9.

2.7 Conclusion

Based on Freyd covers of arbitrary appropriate categories of domains (subsuming all the well known examples) we have defined a class of models for intensional type theory which allow to refute most of the sequents which are provable in extensional ICST but are not derivable in intensional ICST.
All these models are semi-intensional in the sense that they validate the sequent

$$A : \mathrm{Set}\, ,\, B : A \to \mathrm{Set}\, ,\, x : A\, ,\, y : A\, ,\, z : \mathrm{Id}\ A\ x\ y\ \vdash B(x)\ =\ B(y) \in \mathrm{Set}$$

which is also assumed as an axiom in the type theory which was studied by J. Smith in his Thesis (cf. [Smi]) at the end of the seventies.
Although he used a variant without an ambient logical framework his work is related to our semantics for Set in the sense that he interpreted sets as *predicates* on an axiomatically given algorithmic universe whereas we interpret sets as *arbitrary subsets* of the fixed algorithmic universe D .
Both his and our approach are based on the seminal work of P.Aczel (cf. [Acz]). The work of J. Smith uses the method of interpreting type theory by translating it to another constructive theory (a type-free logical theory of constructions) whereas we have chosen a purely semantical, model-theoretic approach.
What is *new* in our approach is that *we can also interpret the ambient logical framework.*
Technically speaking, this has been achieved by shifting to the more complex category IRS of intensional realizability sets whereas the model-theoretic counterpart of J. Smith's account stays within the full subcategory of those intensional realizability sets $\underline{X}$ where $\mathrm{Dom}(\underline{X}) = D$ and $r(\underline{X})$ is injective.

The drawback of being only semi-intensional will be remedied in the next Chapter 3 where we introduce a model which allows to refute the crucial sequent above and furthermore arises as a slight variation of the well-known realizability model for extensional type theory as studied e.g. in [Str1].

Chapter 3

Fully Intensional Models of Constructive Set Theory

In this chapter we describe and discuss a fully intensional model of ICST which also refutes the semi-extensional sequents which were still valid in the models of Chapter 2. A pleasant aspect of these fully intensional models is that they are obtained by only a slight modification from the well known realizability models for extensional type theory based on Kleene realizability.

In section 1 we introduce and motivate the conceptual distinction between potential and actual objects. We argue that in order to avoid semi-extensional phenomena in (realizability) models one must shift the distinction between potential and actual from the level of realizers to the level of objects of the underlying set.

In section 2 we define the category mr-**Set** of so-called *modified realizability sets* from the category r-**Set** of ordinary realizability sets. Objects of mr-**Set** are objects of r-**Set** together with a distinguished chosen subset of actual objects and morphisms are required to preserve actual objects. We discuss the categorical properties of mr-**Set** and find that they are sufficient to interpret extensional type theory and therefore the ambient logical framework. We further show that mr-**Set** appears as the full subcategory of $\neg$closed objects of an appropriate realizability topos.

In section 3 we identify a small full subcategory of mr-**Set** which is internally closed under arbitrary products and will be used for interpreting impredicative ICST. In section 4 we interpret Martin-Löf's identity sets together with our new eliminator K in this (internal) subcategory and show that the resulting model is fully intensional. In section 5 we give intensional interpretations of dependent product and sum types.

In section 6 we give intensional interpretations of N , the empty set and the singleton set and demonstrate that a lot of properties valid in the extensional theory fail in the model and therefore are not derivable in ICST. In section 7 we show that again extensionality principles for functions are refuted by the model.

In section 8 we study the relationship between Martin-Löf's identity sets and Leibniz equality. We show that both notions of equality - although logically equivalent - are not isomorphic. Furthermore we show that Leibniz equality sets cannot be endowed with an eliminator J in our model - and therefore not uniformly by a syntactic definition - as the interpretations of both notions of equality are qualitatively different in a 'topological' sense.

In section 8 we define another small full subcategory of mr-**Set** internally closed under arbitrary products whose underlying realizabilty sets correspond to effective domains. For this new model of Set we show that dependent sums cannot be interpreted.

Finally in section 8 we discuss the relevance of our models for obtaining independence results and checking the adequacy of suggested extensions of ICST.

Chapter 3

3.1 Full Intensionality and Actual vs. Potential Objects

In this chapter we shall study a model of ICST which satisfies the following three

Criteria of Intensionality

(C1) it is *not valid* that

$$A : \mathrm{Set}\ ,\ x : A\ ,\ y : A\ ,\ z : \mathrm{Id}\ A\ x\ y\ \vdash x = y \in A$$

(C2) it is *not valid* that

$$A : \mathrm{Set}\ ,\ B : A \to \mathrm{Set}\ ,\ x : A\ ,\ y : A\ ,\ z : \mathrm{Id}\ A\ x\ y\ \vdash\ B(x) = B(y) \in \mathrm{Set}$$

(C3) for any set A

$$\vdash p \in \mathrm{Id}\ A\ t\ s \quad \textit{implies} \quad \vdash t = s \in A \quad .$$

Criteria (C1) and (C3) are already satisfied in the models presented in Chapter 2 , but criterion (C2), obviously is not satisfied as identity types are either empty or singleton and Set carries a trivial realizability structure (there is only one potential realizer).
Thus two morphisms $B\ ,\ C : A \to \mathrm{Set}$ are equal iff their underlying functions are equal.
A first attempt to avoid this drawback would be to look for a nontrivial realizability structure on Set , but that seems to be fairly impossible as that would amount to introducing *codes* for sets and then we already would be quite close to term models which we want to avoid as they are quite untractable due to their purely syntactical nature.

Therefore one needs a new idea of how to capture on the semantical level the very idea of intensionality. Our approach is based on an adaptation of Kreisel's idea of *modified realizability* with its distinction between *potential* and *actual* objects. (This idea has come up again quite recently in the work of Burstall and McKinna on the so called *deliverables*, see e.g. [McK]).

We first will give an intuitive explanation of the idea behind the distinction between potential and actual objects.

Chapter 3

According to ancient tradition in logic proving a universal statement of the form $(\forall x{:}A)\, P(x)$ amounts to demonstrating that $P(a)$ holds for the *"most general object* a *of type* A". Of course, the most general object is different from all concrete, i.e. most specific, objects. Typically, if A is the type of natural numbers then, of course, any specific natural number is different from the most general natural number as it has a property not shared by all natural numbers. Of course, if predicate P is true for the most general object of type A then it must be true for all concrete objects of type A *but* there is no evidence that if P is valid for all concrete objects of type A that it must be necessarily the case that P is true also for the most general object of type A.
Thus a proposition of the form $(\forall x{:}A)\, P(x)$ may be true individually for every concrete object of type A although it need not be true for the most general object of type A.
Of course, the idea of a most general object (of a certain type) seems to arise from a confusion between syntax and semantics. In contemporary formulations of logic, e.g. in natural deduction style, this idea of a most general object is reflected on the formal level by the variable conditions in the introduction rule for the universal quantifier : in order to prove the sequent $\Gamma \vdash (\forall x{:}A)\, P(x)$ it is sufficient to prove $\Gamma \vdash P(x)$ where x does not occur freely in Γ, i.e. in Γ there are made no assumptions about the most general object as referred to by x.

Although this idea of a most general object of a certain type is quite vague and cannot be made precise according to standards of modern mathematical logic, it motivates the following point of view : validity of $(\forall x{:}A)\, P(x)$ means more than validity of $P(c)$ for all concrete object c of type A.
Therefore it seems to be natural to assume that besides the *actual concrete* objects there are also *potential ideal* objects for which the predicate P has to hold as well in order to be universally valid.
For motivation just consider the situation one has e.g. in nonstandard models of Peano arithmetic. If a universal statement $(\forall x{:}N)\, P(x)$ is true in the standard model but not derivable from the axioms of Peano arithmetic then there exists a nonstandard model of Peano arithmetic containing an element which is not denotable by a term, i.e. a so-called nonstandard element, for which the predicate P does not hold. Thus semantically speaking according to the completeness theorem a proposition $(\forall x{:}A)\, P(x)$ is derivable or universally valid iff for all models (in our case of Peano arithmetic) the predicate is not only true for standard elements but also for all nonstandard elements.
Therefore we consider a type A to be a collection of *potential* or *hypothetical* objects together with a distinguished subcollection of *actual* or *concrete* objects. Of course, a function from a type A to a type B is assumed to *preserve actual objects*.

As this distinction between actual and potential objects is not present in the well-known Kleene realizability models for type theory we will next give an extension of this notion of model in a way that in these extended models there will be such a distinction.

3.2 The Category mr-Set of Modified Realizability Sets

We assume to be known the *category* **r-Set** *of realizability sets*, see e.g. [Str1,2], where it was called **ω-Set**. We adopt to this new notation as it is in accordance with the notation **mr-Set** for the category of *modified realizability sets* which will be defined next.

Definition 3.1

A *modified realizability set* (mr-set) is a triple $\underline{X} = (X , X_0 , \Vdash_X)$ such that

(i) $\mathfrak{U}(\underline{X}) = (X , \Vdash_X)$ is an r-set

(ii) X_0 is a subset of X .

The objects of X are called the *potential* elements and the objects of the subset X_0 are called *actual, defined* or *standard.*

A *morphism of modified realizability sets* from $\underline{X} = (X , X_0 , \Vdash_X)$ to $\underline{Y} = (Y , Y_0 , \Vdash_Y)$ is a (set-theortic) function $f : X \to Y$ such that

(i) f is a morphism from $\mathfrak{U}(\underline{X})$ to $\mathfrak{U}(\underline{Y})$ in **r-Set**

(ii) f preserves actual objects, i.e. $x \in X_0$ implies $f(x) \in Y_0$.

We write **mr-Set** for the category whose objects are modified realizability sets and whose morphisms are morphism of modified realizability sets. ◊

For the more categorically inclined reader we just remark that the category **mr-Set** can be obtained by a rather simple categorical construction : consider regular subobjects in **r-Set** fibred over **r-Set**. Then the total category of this fibration is just **mr-Set**. Actually, this fibration is given by the forgetful functor $\mathfrak{U} :$ **mr-Set** $\to$ **r-Set**.
It seems to be remarkable that this functor has both a left adjoint $\mathfrak{L}$ and a right adjoint $\mathfrak{R}$ which are both right inverses of $\mathfrak{U}$. These functors $\mathfrak{L}$ and $\mathfrak{R}$ are defined as follows : for an r-set $\underline{X} = (X , \Vdash_X)$ we have $\mathfrak{L}\,\underline{X} = (X , \{\} , \Vdash_X)$ and $\mathfrak{R}\,\underline{X} = (X , X , \Vdash_X)$ and for a morphism f in **r-Set** we have $\mathfrak{L}(f) = f = \mathfrak{R}(f)$.
Thus **r-Set** appears as a full subcategory of **mr-Set** in a rather strong way usually called *essential localization* or UIO (Unity and Identity of Opposites), see [Law]. That means that **r-Set** appears as a full subcategory of **mr-Set** in two different ways : once *negatively* as the image under $\mathfrak{L}$ as the full subcategory of **mr-Set** consisting of all those mr-sets *having no actual objects* and once *positively* as the image under $\mathfrak{R}$ as the full subcategory of **mr-Set** consisting of all those mr-sets *where all potential objects are also actual.* Of course, the posi-

tive and the negative appearance of r-**Set** in mr-**Set** have a trivial intersection which contains exactly the initial object $(\{\},\{\},\{\})$ of mr-**Set**. Of course, $\mathfrak{L}$(r-**Set**) is a coreflective subcategory of mr-**Set** and $\mathfrak{R}$(r-**Set**) is a reflective subcategory of mr-**Set**. For any r-set $\underline{X} = (X, \Vdash_X)$ the fibre $\mathfrak{U}_{\underline{X}}$ (i.e. the set of all mr-sets mapped to $\underline{X}$ by $\mathfrak{U}$) is in one-to-one correspondence with the powerset $\mathfrak{P}(X)$ ordered by set inclusion. This fibre $\mathfrak{U}_{\underline{X}}$ is a complete lattice whose elements can be understood as degrees of actuallness describing how big a part of potential objects has become actual.

Before describing the logical properties of the categor mr-**Set** we describe how it can be obtained as the *full subcategory of double negation separated objects of some realizability topos*. This realizability topos arises - by the well known construction of a topos from a tripos, see [HJP] - from the following tripos based on **Set**.

A proposition is a pair (P, p) s.t. $p \subseteq P \subseteq \omega$ and if p is non-empty then $p = P$. An object in the fibre over I is a family $(P_i, p_i)_{i\in I}$ s.t. (P_i, p_i) is a proposition for all $i \in I$.
Given objects $(P_i, p_i)_{i\in I}$ and $(Q_i, q_i)_{i\in I}$ in the fibre over I then

$(P_i, p_i)_{i\in I} \vdash_I (Q_i, q_i)_{i\in I}$

iff

there exists $n \in \omega$ s.t. for all $i \in I$:

if $m \in P_i$ then $\{n\}(m)$ is defined and $\{n\}(m) \in Q_i$

and

if $m \in p_i$ then $\{n\}(m)$ is defined and $\{n\}(m) \in q_i$

A generic object for this hyperdoctrine is given by the identity function on the set of all propositions (as described above).

This tripos construction can be understood as the glueing of Kleene realizability and Boolean logic as it is isomorphic to the tripos over **Set** where propositions are pairs (P, b) with $P \subseteq \omega$ and $b \in \text{Bool} = \{\text{true}, \text{false}\}$ such that $M = \{\}$ implies $b = \text{false}$. This intuitively means that if a proposition (M, b) is true, i.e. $b = \text{true}$, then it must be realizable, i.e. M is non-empty. Thus truth is more restrictive than realizability, i.e. there are false propositions which are realized.
Notice that this tripos construction is dual to van Oosten's notion of q-realizability, see [vO]. There propositions are defined as pairs (M, b) with $P \subseteq \omega$ and $b \in \text{Bool} = \{\text{true}, \text{false}\}$ s.t. $b = \text{false}$ implies $M = \{\}$. This condition intuitively means that if a proposition (M, b)

is realizable, i.e. M is non-empty, then it must be true, i.e. $b = \text{true}$. In this case one claims that realizable propositions are true and admits propositions which are true but not realizable.

We want to emphasize that our category mr-**Set** is not equivalent to the full subcategory of double negation separated objects of Hyland's modified realizability topos **Mod** , see [vO]. The full subcategory of separated objects of **Mod** is equivalent to the category which can be described quite elementarily as follows :

objects are pairs $(\underline{X}, A)$ with $\underline{X}$ an r-set and $A \subseteq \omega$

s.t. $0 \in A$ and $n \Vdash_X x$ implies $n \in A$ for all $x \in X$

morphisms from $(\underline{X}, A)$ to $(\underline{Y}, B)$ are set-theoretic functions $f : X \to Y$

for which there exists $n \in \omega$ s.t.

n realizes f as a morphism from $\underline{X}$ to $\underline{Y}$ in mr-**Set**

and

for any $m \in A$: $\{n\}(m)$ is defined and $\{n\}(m) \in B$

This category does not serve our purposes as there is no distinction between actual and potential objects but only a distinction between actual and potential realizers.
But, of course, the idea behind Hyland's definition of **Mod** is a different one, namely to build a topos from a tripos whose propositions are propositions in the sense of Kreisel's modified realizability with the typical distiction between actual and potential realizers. The propositions of this underlying tripos are pairs (P, p) with $p \subseteq P \subseteq \omega$ and $0 \in P$. Of course, *the propositions of this tripos do not appear as objects of the modified realizability topos.*
In contrast the intention of our definition of mr-**Set** is to define a category where *the objects themselves appear as propositions in the sense of modified realizability.*
Anyway, we do not consider it to be too important that mr-**Set** appears as the category of double negation separated objects for some realizability topos as we consider type theory as more fundamental than topos logic.

Next we will prove some important categorical properties of mr-**Set** guaranteeing that it is a model of quite a strong extensional type theory.

Theorem 3.2

The category mr-**Set** is *locally cartesian closed* (lcc) and *regular.*

Furthermore the forgetful functor $\mathfrak{U}$: **mr-Set** $\rightarrow$ **r-Set** preserves the structures of locally-cartesian-closedness and regularity.
The terminal object is not a generator, but the object $(\{*\}, \{\}, \{(0, *)\})$ is a generator.

Proof : First we show that **mr-Set** is left exact, i.e. has all finite limits.

Obviously, in **mr-Set** $(\{*\}, \{*\}, \{(0, *)\})$ is a terminal object.
If $f : \underline{X} \rightarrow \underline{Z}$ and $g : \underline{Y} \rightarrow \underline{Z}$ are morphisms in **mr-Set** then their pullback exists in **mr-Set** and is decribed as follows.
The top of the limiting cone is $\underline{P} = (P, P_0, \Vdash_P)$ where $P = \{(x, y) \mid x \in X, y \in Y$ and $f(x) = g(y)\}$, $P_0 = \{(x, y) \in P \mid x \in X_0$ and $y \in Y_0\}$ and $n \Vdash_P (x, y)$ iff $\pi_0(n) \Vdash_X x$ and $\pi_1(n) \Vdash_Y y$.
The edges of the limiting cone are given by the morphisms $p : \underline{P} \rightarrow \underline{X}$ and $q : \underline{P} \rightarrow \underline{Y}$ which project objects of P on their first and second component, respectively. Obviously, p and q are realizable and preserve definedness of objects.

It remains to describe the most important part of the lcc-structure : the right adjoints to pullback functors. If $f : \underline{Y} \rightarrow \underline{X}$ and $g : \underline{Z} \rightarrow \underline{Y}$ are morphisms in **mr-Set** then the morphism $\Pi_f(g) : \underline{P} \rightarrow \underline{X}$ is described as follows. The object $\underline{P} = (P, P_0, \Vdash_P)$ consists of

$P = \{ (x, s) \mid \quad x \in X$ and $s : f^{-1}(\{x\}) \rightarrow Z$ such that

$g(s(y)) = y$ for all $y \in Y$ with $f(y) = x$

and

there exists $n \in \omega$ such that for all $y \in Y$ with $f(y) = x$

and $m \in \omega$ with $m \Vdash_Y y$: $\{n\}(m)$ terminates and $\{n\}(m) \Vdash_Z s(y) \}$

and P_0 is that subset of P containing exactly those pairs where the first component is defined and the second component preserves definedness, i.e. for all $(x, s) \in P$ we have

$(x, s) \in P_0$ iff $x \in X_0$ and $s(y) \in Z_0$ for all $y \in Y_0$ with $f(y) = x$

and

$n \Vdash_P (x, s)$ iff $\pi_0(n) \Vdash_X x$ and

for all $y \in Y$ with $f(y) = x$ and $m \in \omega$ with $m \Vdash_Y y$:
$\pi_0(n)\}(m)$ terminates and $\{\pi_0(n)\}(m) \Vdash_Z s(y)$

The morphism $\Pi_f(g) : \underline{P} \to \underline{X}$ is given by the projection on the first component of pairs in P which, obviously, is realizable and preseves definedness of objects.

Finally the counit of the adjunction $f^* \dashv \Pi_f$ is given for any morphism g with codomain $\underline{Y}$ by the *evaluation morphism* $ev_f(g) : f^*\Pi_f(g) \to g$ defined as follows :

$$ev_f(g)\,(\,y\,,(\,x\,,p\,)\,))\;=\;s(y) \quad \text{for any } (\,x\,,s\,) \in P \text{ and } y \in f^{-1}(\{x\})$$

which, obviously, is realizable and preserves definedness.

In order to show regularity consider an arbitrary morphism $f : \underline{X} \to \underline{Y}$ in **mr-Set** .
Then the initial epi-mono factorization is constructed as follows : let $\underline{I} = (\,I\,, I_0\,, \Vdash_I\,)$ where

$$I = \mathrm{im}(f) = \{\; y \in Y \mid y = f(x) \text{ for some } x \in X \;\}$$

$$I_0 = \{\; y \in Y \mid y = f(x) \text{ for some } x \in X_0 \;\}$$

$$n \Vdash_I y \text{ iff } n \Vdash_X x \text{ for some } x \in X \text{ with } y = f(x)\,.$$

The epi e is given by $e(x) = f(x)$ and realized by $\Lambda n.n$ and the inclusion m is given by $m(y) = y$ and realized by any realizer for f . It is straightforward to show that these initial epi-mono factorizations are preserved up to isomorphism by pullbacks along arbitrary morphisms in mr-**Set**, see [Str1].
Looking at the constructions described above and comparing them with the analogous constructions for r-**Set** (see e.g. [Str1]) shows that they simply extend the constructions for r-**Set** by specifiying which objects are defined.

Of course, by the terminal object $(\,\{*\}, \{*\}, \{\,(\,0\,,\; *\,)\,\}\,)$ one can séparate only those morphism which are different at some actual argument because any morphism s from the terminal object to some mr-set $\underline{X}$ must map $*$ to an object in X_0 , i.e. a defined object.
On the other hand the mr-set $(\,\{*\}\,, \{\}\,, \{\,(\,0\,,\; *\,)\,\}\,)$ clearly is a separating object : if f , g : $\underline{X} \to \underline{Y}$ are distinct morphisms then for some $x \in X$ (but not necessarily $x \in X_0$) we have $f(x) \neq g(x)$ and therefore $f \circ s \neq g \circ s$ where $s(*) = x$ and s is well defined as $*$ is not defined and therefore x need not be defined. $\Diamond$

Thus the category mr-**Set** in fact is a model for *extensional* Martin-Löf's type theory and therfore provides a model for a Logical Framework.
We have not yet achieved our goal of providing a fully intensional model of Impredicative Martin-Löf type theory. This will be done on the next paragraph.

3.3. A Small Full Subcategory of mr-Set Closed Under Arbitrary Products

As indicated aboce the way we proceed in order to build a fully intensional model for impredicative Martin-Löf type theory is to take **mr-Set** as a model for the ambient Logical Framework - which has to be extensional , see e.g. [NPS] - and then to define a *small full internal category of sets* which is *closed under arbitrary products* and *on which level all the intensional features will show up*.
The paradigmatic way of how to find small full internal categories in **mr-Set** which are internally closed under arbitrary products is to start from such a category inside **r-Set** and then to perform again the modified realizability construction. Of course, there is an abundancy of such categories in **r-Set** , see e.g. [Str1].
We will first explain this construction in some generality but then concentrate on the examples arising from the following two small full internal categories in **r-Set** : the category PER-0 of all partial equivalence relations R on ω such that $0 \, R \, 0$ and the category $\underline{\text{CExPer}^{\perp}}$ of complete extensional pers with bottom (realized by any code for the partial recursive function which nowhere terminates) as defined in [FMRS]. The characteristic feature of these categories is that they are all internally closed under arbitrary products *but not under equalizers*. This latter condition is important in order to avoid the existence of extensional identity types.
If one chooses any of these categories as interpretations of the type Set (where the ambient logical framework is interpreted as **r-Set**) then one gets a model of intensional constructive set theory where it is impossible to interpret extensional identity types (due to the lack of empty types) but there is an interpretation of intensional identity types satisfying the criteria (C1) and (C2) but *not* the essential criterion (C3) (cf. Chapter 2 , §2.1).

In all these categories *any type is inhabited* which guarantees the validity of criteria (C1) and (C2). But exactly this property makes (C3) fail as any type and therefore also any any identity type Id A t s is inhabited even if it does not hold that $t = s \in A$!
But if we choose **mr-Set** instead of simply **r-Set** for the interpretation of the ambient Logical Framework the we can have the benefits of both emptyness and non-emptyness at the same time, as there are types which at the same time are *nonempty w.r.t. potential objects* and *empty w.r.t. actual objects*. The fact that in *some sense types can be empty and nonempty at the same time* has been the essential motivation for us to change from **r-Set** to **mr-Set** .

Next we will describe how to lift the full internally complete subcategory PER of **r-Set** to a full internally complete subcategory of **mr-Set** .

Definition 3.3

Let $\underline{\text{Set}} = (\text{Set}, \text{Set}_0, \Vdash_{\text{Set}})$ be the mr-set where

$$\mathrm{Set} = \{ (R, P) \mid R \in \mathrm{PER} \text{ and } P \subseteq \omega/R \}$$

$$\mathrm{Set}_0 = \mathrm{Set}$$

$$\Vdash_{\mathrm{Set}} = \omega \times \mathrm{Set} \quad .$$

Let $\underline{\mathrm{El}} = (\mathrm{El}, \mathrm{El}_0, \Vdash_{\mathrm{El}})$ be the mr-set where

$$\mathrm{El} = \{ (R, P, M) \mid (R, P) \in \mathrm{Set} \text{ and } M \in \omega/R \}$$

$$\mathrm{El}_0 = \{ (R, P, M) \in \mathrm{Set} \mid M \in P \}$$

$$\Vdash_{\mathrm{El}} = \{ (n, (R, P, M)) \mid (R, P, M) \in \mathrm{El} \text{ and } n \in M \} \quad .$$

Let ext : $\underline{\mathrm{El}} \to \underline{\mathrm{Set}}$ be the morphism in mr-**Set** with ext $(R, P, M) = (R, P)$.

We call a *family of sets* any morphism which can be obtained as a pullback of ext along some morphism in mr-**Set** and let $\mathfrak{S}$ denote the class of all families of sets. ◊

From Theorem 3.2 and the facts we know about PER , see e.g. [Str1] , the following Theorem follows immediately.

Theorem 3.4 .

The class $\mathfrak{S}$ of display maps represents a small full internally complete subcategory of r-**Set** where the class $\mathfrak{S}$ is closed under composition.
In the full subcategory as given by $\mathfrak{S}$ it is still possible to interpret extensional identity types because the full subcategory is closed under equalizers. ◊

Therefore in order to obtain fully intensional models we have to restrict the subcategory given by $\mathfrak{S}$ to subcategories which are *closed under internal products but not under equalizers* and *still allow to interpret intensional identity types.*

For this purpose we need the following auxiliary definition.

Definition 3.5

For any subset $\mathfrak{Q}$ of PER let $\underline{\mathrm{Set}}(\mathfrak{Q})$ denote the regular subobject of $\underline{\mathrm{Set}}$ given by

$$\mathrm{Set}(\mathfrak{Q}) = \{ (R, P) \in \mathrm{Set} \mid R \in \mathfrak{Q} \} = \mathrm{Set}_0(\mathfrak{Q})$$

Let $\mathfrak{S}(\mathfrak{Q})$ denote the class of all morphisms which can be obtained by pulling back the generic morphism $\mathrm{ext} : \underline{\mathrm{El}} \to \underline{\mathrm{Set}}$ along some morphism which factors through the inclusion of $\underline{\mathrm{Set}}(\mathfrak{Q})$ into $\underline{\mathrm{Set}}$.

Let $\underline{\mathrm{El}}(\mathfrak{Q}) = (\mathrm{El}(\mathfrak{Q}), \mathrm{El}_0(\mathfrak{Q}), \Vdash_{\mathrm{El}(\mathfrak{Q})})$ be the mr-set where

$$\mathrm{El}(\mathfrak{Q}) = \{ (R, P, M) \mid (R, P) \in \mathrm{El} \text{ and } R \in \mathfrak{Q} \}$$

$$\mathrm{El}_0(\mathfrak{Q}) = \{ (R, P, M) \in \mathrm{El}(\mathfrak{Q}) \mid M \in P \}$$

$$\Vdash_{\mathrm{El}(\mathfrak{Q})} = \{ (n, (R, P, M)) \mid (R, P, M) \in \mathrm{El}(\mathfrak{Q}) \text{ and } n \in M \} .$$

Let $\mathrm{ext}(\mathfrak{Q}) : \underline{\mathrm{El}}(\mathfrak{Q}) \to \underline{\mathrm{Set}}(\mathfrak{Q})$ be the morphism in mr-**Set** with $\mathrm{ext}(R, P, M) = (R, P)$ which, obviously, is a generic morphism for $\mathfrak{S}(\mathfrak{Q})$. $\Diamond$

The first and most important model we will study is obtained by choosing for $\mathfrak{Q}$ the collection of all partial equivalence relations containing 0 in its carrier.

In order to guarantee closure under all internal products for the rest of this chapter we make the following

Assumption about Goedel Numberings

$\{0\}(n) = 0$ for all $n \in \omega$

$< 0, 0 > = 0$ $\Diamond$

Actually this assumption is a very mild restriction as $< 0, 0 > = 0$ is satisfied anyway for the standard bijection between natural numbers and pairs of natural numbers and addmissible Goedel numberings for the partial recursive functions are unique up to isomorphism (see e.g. [Ro]) and clearly there is one such in the whole isomorphism class satisfying the claim that 0 is a code for the constant zero function.

Definition 3.6

Let $\mathrm{PER\text{-}0} = \{ R \in \mathrm{PER} \mid 0 \mathrel{R} 0 \}$ be the collection of all pers having 0 in its carrier.

We introduce the following abbreviations :

$\underline{\text{Set}}_1 = \underline{\text{Set}}(\text{PER-0})$

$\underline{\text{El}}_1 = \underline{\text{El}}(\text{PER-0})$

$\mathfrak{S}_1 = \mathfrak{S}(\text{PER-0})$

$\text{ext}_1 = \text{ext}(\text{PER-0})$. ◊

Theorem 3.7

The full subcategory of mr-**Set** represented by $\mathfrak{S}_1$ is internally closed under arbitrary products but fibrewise diagonals of non-monic display maps are not contained in $\mathfrak{S}_1$ and therefore extensional identity types cannot be interpreted.

Proof : Given any family F of pers in PER-0 indexed over an r-set $\underline{X}$ then the product of this family contains an object realized by 0 : the function f which maps any $x \in X$ to the unique object of F(x) realized by 0 .
Any display map in $\mathfrak{S}_1$ is an epimorphism as the underlying map always is surjective because any fibre contains at least one element realized by 0 . But on the other hand for any display map a which is not monic the equalizer of the kernel pair of a is not epic and therefore cannot be in $\mathfrak{S}_1$. ◊

Next we will show that intensional identity types can be interpreted very naturally in $\underline{\text{Set}}_1$.

In order to interpret the logical framework in mr-**Set** *and* ICST *in the full subcategory given by* ext_1 *it would be necessary to turn the whole structure into a contextual category of the appropriate form. For the case of* r-**Set** *this has been performed in all details in* [Str1]. *But as these details do not add to a better understanding when showing that a model is fully intensional we prefer to work informally, but rigorously, in the model described above and the other ones we will employ in this chapter.*

We now turn to the interpretation of intensional identity types in $\underline{\text{Set}}_1$.

Definition 3.8

For any set $(R, P) \in \text{Set}_1$ and $x, y \in \omega/R$ we define $\text{Id}(R, P)\, x\, y \in \text{Set}_1$ according to the following case analysis :

- if $x = y$ then

$$Id(R,P)\,x\,y = (\{(0,0),(1,1)\},\{\{1\}\})$$

- if $x \neq y$ then

$$Id(R,P)\,x\,y = (\{(0,0)\},\{\})$$

i.e. if $x = y$ the corresponding identity set contains the actual object $\{1\}$ representing the actual proof object of the set expressing the propositional equality of x and y and the non-actual, only potential object $\{0\}$ and if $x \neq y$ then the corresponding identity set contains only the potential, non-actual object $\{0\}$.

The constructor r is interpreted as follows : for any set $(R,P) \in Set_1$ and $x \in \omega/R$

$$r(R,P)\,x = \{1\} \in Id(R,P)\,x\,x$$

and, obviously, the function r is realizable and it trivially preserves definedness as the result $\{1\} \in Id(R,P)\,x\,x$ is always defined.

The eliminator J is defined as follows : given $(R,P) \in Set_1$, a family C of sets, i.e. objects in Set_1 indexed over $x, y \in \omega/R$ and $z \in Id(R,P)\,x\,y$ and a realizable mapping d which for any $x \in \omega/R$ chooses an object $d\,x \in C\,x\,x\,\{1\}$ then $J(R,P)\,C\,d$ is defined by the following case analysis :

- for $x \in \omega/R$

$$J(R,P)\,C\,d\,x\,x\,\{1\} = d\,x$$

- for $x, y \in \omega/R$

$$J(R,P)\,C\,d\,x\,y\,\{0\} = \text{the unique object of } C\,x\,y\,\{0\} \text{ realized by } 0$$

The morphism $J(R,P)\,C\,d$ is realized e.g. by the algorithm which looks at the realizer for the last argument (which is of type $Id(R,P)\,x\,y$ and therefore it can be realized only by 0 or 1) and in the case that it is 1 applies the realizer of d to the realizer of x , and in the case that it is 0 terminates with value 0 and in all other cases diverges.
The preservation of definedness is guaranteed because if the third argument is $\{0\}$ then it is undefined and therefore there are no constraints on the output and if the the third argument is $\{1\}$ then $J(R,P)\,C\,d\,x\,x\,\{1\} = d\,x$ is defined if x is defined under the assumption that d

is defined, i.e. maps defined objects to defined objects.

Similarly one can interpret the eliminator K : given $(R, P) \in \mathrm{Set}_1$, a family C of sets, i.e. objects in Set_1 indexed over $x \in \omega/R$, and $z \in \mathrm{Id}(R, P)\, x\, x$ and a realizable mapping d which for any $x \in \omega/R$ chooses an object $d\, x \in C\, x\, \{1\}$ then $K(R, P)\, C\, d$ is defined by the following case analysis :

- $K(R, P)\, C\, d\, x\, \{1\} = d\, x$

- $K(R, P)\, C\, d\, x\, \{0\} =$ the unique object of $C\, x\, \{0\}$ realized by 0

for all $x \in \omega/R$. ◊

After having given the interpretation of identity sets we are ready to prove that this model is fully intensional.

3.4. The Model Given By $\mathfrak{S}_1$ Is Fully Intensional

Next we shall prove that this interpretation of intensional identity types satisfies the three criteria of intensionality mentioned at the beginning of this chapter. We moreover provide *more specific* examples of sequents (making heavily use of identity sets) which allow to refute the condition (C1) - (C3).

Theorem 3.9

In the model given by $\mathfrak{S}_1$ the three criteria (C1) - (C3) are satisfied.

Furthermore we have that

(1) For any set A the sequent

$$a : A,\ x : \mathrm{Id}\, A\, a\, a,\ y : \mathrm{Id}\, A\, a\, a,\ z : \mathrm{Id}\, (\mathrm{Id}\, A\, a\, a)\, x\, y \ \vdash\ x = y \in \mathrm{Id}\, A\, a\, a$$

is not valid in the model and whenever a is an object in A then the sequent

$$x : \mathrm{Id}\, A\, a\, a,\ y : \mathrm{Id}\, A\, a\, a,\ z : \mathrm{Id}\, (\mathrm{Id}\, A\, a\, a)\, x\, y \ \vdash\ x = y \in \mathrm{Id}\, A\, a\, a$$

is not valid in the model.

(2) If A is a set then

$$a : A\ ,\ x : \mathrm{Id}\ A\ a\ a\ ,\ y : \mathrm{Id}\ A\ a\ a\ ,\ z : \mathrm{Id}\ (\mathrm{Id}\ A\ a\ a)\ x\ y$$
$$\vdash$$
$$\mathrm{Id}\ (\mathrm{Id}\ A\ a\ a)\ (r\ A\ a)\ x = \mathrm{Id}\ (\mathrm{Id}\ A\ a\ a)\ (r\ A\ a)\ y \ \in\ \mathrm{Set}$$

is not valid in the model and if a is an object of A then the sequent

$$x : \mathrm{Id}\ A\ a\ a\ ,\ y : \mathrm{Id}\ A\ a\ a\ ,\ z : \mathrm{Id}\ (\mathrm{Id}\ A\ a\ a)\ x\ y$$
$$\vdash$$
$$\mathrm{Id}\ (\mathrm{Id}\ A\ a\ a)\ (r\ A\ a)\ x = \mathrm{Id}\ (\mathrm{Id}\ A\ a\ a)\ (r\ A\ a)\ y \ \in\ \mathrm{Set}$$

is not valid in the model.

Proof : The criterion (C1) is fulfilled as the sequent

$$A : \mathrm{Set}\ ,\ x : A\ ,\ y : A\ ,\ z : \mathrm{Id}\ A\ x\ y\ \vdash\ x = y \in A$$

is not valid as if we choose for A a set containing at least two different (potential) objects x and y and for z the potential object $\{0\}$ then it does not hold that $x = y$. Here, of course, it is important that *morphisms must be equal also at non-actual arguments*.

Whenever A is a set and a is a potential object of A (there always must exist at least one !) the set Id A a a contains the two different objects $\{0\}$ (potential) and $\{1\}$. (actual) . Therefore the sequent

$$x : \mathrm{Id}\ A\ a\ a\ ,\ y : \mathrm{Id}\ A\ a\ a\ ,\ z : \mathrm{Id}\ (\mathrm{Id}\ A\ a\ a)\ x\ y\ \vdash\ x = y \in\ \mathrm{Id}\ A\ a\ a$$

is not valid in the model. This proves our claim (1).

The criterion (C2) is fulfilled as the sequent

$$A : \mathrm{Set}\ ,\ B : A \rightarrow \mathrm{Set}\ ,\ x : A\ ,\ y : A\ ,\ z : \mathrm{Id}\ A\ x\ y\ \vdash\ B(x) = B(y) \in\ \mathrm{Set}$$

is not valid as if we choose for A a set containing at least two different (potential) objects x and y and for z the potential object $\{0\}$ and for B a family of sets with $B(x) \neq B(y)$ then the consequence of the sequent is obviously false. The point is that there always exists a family of sets B with $B(x) \neq B(y)$ provided $x \neq y$ as *all* set-theoretic functions from the underlying set of A to Set_1 correspond to a morphism in our model.

For any set A and any object a in A consider the type Id A a a and the family

$$[z\colon \mathrm{Id}\ A\ a\ a]\ \mathrm{Id}\ (\mathrm{Id}\ A\ a\ a)\ (r\ A\ a)\ z \in (\mathrm{Id}\ A\ a\ a) \to \mathrm{Set}$$

Then the sets

$$B(r\ A\ a) = \mathrm{Id}\ (\mathrm{Id}\ A\ a\ a)\ (r\ A\ a)\ (r\ A\ a) \quad \text{and}$$

$$B(\{0\}) = \mathrm{Id}\ (\mathrm{Id}\ A\ a\ a)\ (r\ A\ a)\ \{0\}$$

are obviously different as $r\ A\ a \neq \{0\}$ and therefore the sequent

$$\begin{array}{l} x : \mathrm{Id}\ A\ a\ a\ ,\ y : \mathrm{Id}\ A\ a\ a\ ,\ z : \mathrm{Id}\ (\mathrm{Id}\ A\ a\ a)\ x\ y \\ \vdash \\ \mathrm{Id}\ (\mathrm{Id}\ A\ a\ a)\ (r\ A\ a)\ x \ = \ \mathrm{Id}\ (\mathrm{Id}\ A\ a\ a)\ (r\ A\ a)\ y \ \in \ \mathrm{Set} \end{array}$$

is not valid in the model. This proves our claim (2).

The criterion (C3) is fulfilled as the interpretation of the empty context is the terminal object in mr-**Set** . Suppose that $\vdash p \in \mathrm{Id}\ A\ t\ s$ is valid in the model then the interpretation of t is a morphism f from the terminal object in mr-**Set** to the interpretation of the type $\mathrm{Id}\ A\ t\ s$. As the terminal object is $(\{*\}, \{*\}, \{(0, *)\})$ and f preserves definedness of objects we have that $f(*)$ is a defined object in the interpretation of $\mathrm{Id}\ A\ t\ s$. As there is at most one defined object in $\mathrm{Id}\ A\ t\ s$ it follows that $f(*) = \{1\} \in \mathrm{Id}\ A\ t\ s$ and therefore the interpretations of t and s are equal and therefore the judgement $t = s \in A$ is valid in the model. ◊

Remark. Although the model satisfies our three criteria of intensionality it still validates some sequents which are not derivable syntactically as e.g.

$$A : \mathrm{Set}\ ,\ x : A\ ,\ y : A \ \vdash\ \mathrm{Id}\ A\ x\ y \ = \ \mathrm{Id}\ A\ y\ x \ \in \ \mathrm{Set} \quad .$$

We just indicate how by a slight modification of the definition of identity types our model can be adjusted to refute also the sequent above expressing the symmetry of identity types.
Instead of having just one undefined object in all identity types one might introduce for some identity types another undefined object realized only by the number 2 .
A possible interpretation of intensional identity types refuting symmetry would be the following :

- if $x = y$ then

$$\mathrm{Id}\,(R, P)\ x\ y \ = \ (\{(0,0), (1,1)\}, \{\{1\}\})$$

- if $x \neq y$ and $x \in P$ then

$$Id(R,P)\,x\,y = (\{(0,0),(2,2)\},\{\})$$

- if $x \neq y$ and $x \notin P$ then

$$Id(R,P)\,x\,y = (\{(0,0)\},\{\}) \quad .$$

Then if a and b are different objects of a type A where one of them is actual and the other is only potential then the interpretations of the types Id A a b and Id A b a are different as one contains two potential objects and the other one only one potential object.
The definitions of the constructor r and the eliminators J and K can be extended in a straightforward way (no changes in the definition of r and K are necessary, and, provided $\{2\}$ is a potential object in Id A x y, we put $J\,A\,C\,d\,x\,y\,\{0\} = J\,A\,C\,d\,x\,y\,\{2\}$).
Furthermore under this interpretation the canonical proof of $Id(R,P)\,x\,y \rightarrow Id(R,P)\,y\,x$ is not an isomorphism anymore as the object realized by 2 will be mapped to the object realized by 0.

Anyway we can conclude that if one drops the condition of uniqueness of elimination then interpretations of identity types (in the same model structure) are not unique up to isomorphism anymore.

3.5. Intensional Product and Sum Types

Next we will give the interpretation of the less crucial set forming operations such as Π and Σ. The interesting aspect is that whereas it is impossible to interpret identity sets in an extensional way, seeTheorem 3.6 , it is very well the case that Π and Σ can be interpreted in a way that they satisfy the axioms expressing uniqueness of elimination : namely that the η-.rule and surjective pairing is valid even on the level of judgemental equality. But, as we will show, there is also a lot of non-isomorphic interpretations for Π and Σ (provided that uniqueness of elimination is not claimed for them).

Theorem 3.10

In the model where the type of sets is given by $\underline{Set}_1$ the extensional product of familiy of sets indexed over an arbitrary type is canonically isomorphic to a set and the extensional strong sum of a family of sets indexed over a set is canonically isomorphic to set.
Obviously, under this interpretation the η-rule for λ-abstraction and surjective pairing are valid.

Proof : As 0 codes the function $\Lambda n.0$ and any set contains an object realized by 0 the number 0 realizes that object in the extensional product of a family of sets which maps any argument to the unique object realized by 0 .
As any index set contains an object realized by 0 and the set indexed by this argument also contains an object realized by 0 the strong extensional sum of such a family contains an object realized by 0 as well.
In both cases the constructions give rise to mr-sets where objects are uniquely determined by their realizers and therefore they are isomorphic to (the extension of) a set where both parts of the canonical isomorphism are realized by $\Lambda n.n$. ◊ .

Now we turn to non-extensional interpretations of products and sums of families of sets which are obtained from the extensional ones by adding (discretely) an arbitrary number of nonstandard "error" elements.

Theorem 3.11

In the model where the type of sets is interpreted by $\underline{\text{Set}}_1$ the notions of product and sum of a family of sets can be interpreted in a non-extensional way as follows.

One first interprets the notions extensionally as described in Theorem 3.10 obtaining, say, a per R together with a subset P of ω/R . Then one constructs from this the per S defined as follows :

$$
n\,S\,m \quad \text{iff} \quad \begin{array}{l} n \text{ and } m \text{ are both odd} \\ \text{or} \\ \text{there exist } k \text{ and } l \text{ with } k\,R\,l \text{ and } n = 2k \text{ and } m = 2l . \end{array}
$$

Then one gets an embedding in from R into S which is realized by the algorithm $\Lambda i.2\,i$ and is not surjective as the object realized by an odd number is not in the image.
The subset Q of ω/S is defined as the image of P under the embedding in .

The resulting pair (S , Q) then is the new interpretation.

The new interpretation of constructors is obtained by first interpreting the constructor for the extensional version and afterwards applying in .

The eliminators are interpreted in the following way : given a family C indexed over the object (S , Q) and a function d prescribing the behavior on the image under in then the extension to the whole of (S , Q) is defined by mapping any object which is not in the image of in to the unique object realized by 0 .
For these eliminators the corresponding η-rules fail.

Proof : One only has to show that the elimination is realizable. But as realizers for objects in the image of in can only be even numbers and realizers for the objects in the complement can only be odd numbers the following algorithm realizes the elimination of d where n denotes the realizer for the elimination of d w.r.t. the extensional interpretation : first check whether the realizer m is even or odd, then if n is even apply the realizer n to m/2 and if m is odd then give 0 as result.
As all the objects realized by odd numbers are undefined the elimination of d preserves definedness of objects.
Eliminations are not unique as for a given d prescribing the behaviour on objects realized by even numbers the extension d' of d can be defined for the argument realized by the odd numbers in an arbitrary way. Therefore the η-rules fail. $\diamond$

3.6. Intensional Interpretations of N , N_0 , N_1 and Their Properties

Now let us turn to the intensional interpretation of some non-logical data types. We will show that a lot of sequents - which were already suspected to be underivable in ICST - are actually not valid in out model.

Theorem 3.12

The natural numbers can be interpreted in an extensional way by

$$(\{ (n , n) \mid n \in \omega \} , \{ \{n\} \mid n \in \omega\})$$

A more intensional interpretation is obtained by taking for N

$$(\{ (n , n) \mid n \in \omega \} , \{ \{2 n\} \mid n \in \omega\})$$

where the odd numbers realize the undefined error elements and for any natural number n the even number 2 n realizes the n-th natural number.

Of course, zero is the object realized by 0 .
The successor operation succ is the function realized by $\Lambda n . n + 2$.
Thus we have the ordinary natural number structure *both* on the standard *and* on the nonstandard part.

The eliminator R is defined on the standard part as usual and on the nonstandard part also as usual but with choosing for the initial value of the iteration the unique object realized by 0 in the fibre over $\{1\}$.

The eliminator R does not satisfy the corresponding η-rule which would allow to prove the judgemental equality of functionals by induction.

Proof : We just have to prove that R-elimination is not unique. It follows from the fact that in our definition of R the behavior on the nonstandard part is defined by choosing for the initial value (of the iteration) the unique object realized by 0 in the fibre over $\{1\}$. If there is another (potential) object in the fibre over $\{1\}$ then one can choose this as initial value (of the iteration) and thus obtains another elimination which is different from the one given by R at least at the argument $\{1\}$. ◊

Now we turn to the interpretation of the *singleton set* N_1 .

Theorem 3.13

The singleton set N_1 can be interpreted in an extensional way as

$$(\{ (0,0) \}, \{ \{0\} \}) \quad .$$

A non-extensional interpretation is the set

$$(\{ (0,0), (1,1) \}, \{ \{0\} \})$$

containing one actual object realized by 0 and one only potential object realized by 1 .
The eliminator R_1 assigns to a family C indexed over N_1 and an object d in $C(\{0\})$ a function f mapping the defined object $\{0\}$ of N_1 to the object d and the undefined object $\{1\}$ to the unique object of type $C([1])$ realized by 0 .

The elimination is not unique in general as a function on N_1 is not uniquely determined by its behavior on $\{0\}$. Typically, the elimination is not unique if the type $C(\{1\})$ contains more than one potential object.

If C is a family of sets indexed over N_1 then there is a canonical map f from $C(\{0\})$ to $\{x:N_1\}\ C(x)$ given by the eliminator R_1 and a canonical map g from $\{x:N_1\}\ C(x)$ to $C(\{0\})$ given by evaluation at $\{0\}$. The composition $g \circ f$ is equal to identity (which must hold at it is derivable in the calculus !).
But the composition $f \circ g$ is not judgementally equal to identity if $C(\{1\})$ contains more than one potential object.
If, furthermore, $C(\{1\})$ contains at least one actual object then $f \circ g$ is not even propositio-

nally pointwise equal to identity.

Proof : The function $f \circ g$ is not judgementally equal to identity under the assumption that $C(\{1\})$ contains more than one potential object as for any s in $\{x:N_1\}\ C(x)$ there is a t in $\{x:N_1\}\ C(x)$ such that $s(\{0\}) = t(\{0\})$ but $s(\{1\}) \neq t(\{1\})$.
If, furthermore, one assumes that $C(\{0\})$ contains at least one defined object then the set $\{x:N_1\}\ C(x)$ contains at least one defined object s such that $s(\{1\})$ is different from the unique object realized by 0 and therefore - as $f(g(s))(\{1\})$ is equal to the unique object realized by 0 we have that $s \neq f(g(s))$. Thus $f \circ g$ and the identity are different already at some actual object and therefore the proposition expressing the pointwise propositional equality of $f \circ g$ and the identity does not contain an actual object. ◊

Now we turn to the interpretation of the *empty set* N_0 which in our model admits only an intensional interpretation. The reason is that any set in our model contains at least one potential object and therefore for any set A there exists a set B such that there is more than one morphism from A to B.

Theorem 3.14

The empty set N_0 can be interpreted in our model by

$$(\{(0,0)\},\{\})$$

containing one potential object and no actaul object.

The eliminator R_0 associates with any family C of sets indexed over N_0 the function $R_0\,C$ in $\{x:N_0\}\ C\,x$ mapping $\{0\}$ in N_0 to the unique object realized by 0 in $C(\{0\})$.

Furthermore, for any set $A = (R, P)$ the set $N_0 \rightarrow A$ is isomorphic to the set $(R, \omega/R)$.
Therefore in general A and $N_0 \rightarrow A$ are not isomorphic anymore.
More precisely, A and $N_0 \rightarrow A$ are isomorphic iff all potential objects of A are actual.

The (syntactically definable) function from A to $N_0 \rightarrow A$ mapping A constantly to the object $R_0\,([x:N_0]\ A)$ in $N_0 \rightarrow A$ is an isomorphism if and only if the set A contains exactly one object and this object is actual.

Another possible but non-isomorphic interpretation of N_0 is

$$(\{(n,n) \mid n \in \omega\},\{\})$$

In this case the eliminator R_0 can be defined as follows : if C is a family of sets indexed over N_0 then the canonical extension obtained by the eliminator R_0 is the function which mapping any $\{n\}$ in N_0 to the unique object realized by 0 in $A(\{n\})$.

Under this interpretation N_0 and $N_0 \to A$ are not isomorphic even if all potential objects of A are actual.

Proof : Under the first interpretation of N_0 for any set $A = (R, P)$ the set $N_0 \to A$ is isomorphic to the set $(R, \omega/R)$ because there is an obvious 1-1-correspondence between potential objects of $N_0 \to A$ and potential objects of A . Therefore if all objects of type A are actual then A and $N_0 \to A$ are isomorphic as .
As the only object of N_0 is undefined all potential objects of $N_0 \to A$ are actual (as the maps have nothing to preserve !). Therefore as soon as A contains at least one object which is not actual A and $N_0 \to A$ cannot be isomorphic as in $N_0 \to A$ all objects are actual.

Let f in **r-Set** be the underlying morphism of the function from A to $N_0 \to A$ mapping the set A constantly to $R_0([x{:}N_0]\ A)$. Obviously, f is a constant realizable function. It is an isomorphism if and only if A contains exactly one potential object (as otherwise it would not be onto). But this unique object in A must be actual as all objects in $N_0 \to A$ are actual and f must reflect the property of being actual.

For the second interpretation of the empty type in general N_0 and $N_0 \to A$ are not isomorphic even if A contains only actual objects. Take for A e.g. a set containing only finitely many objects which are all actual then the set $N_0 \to A$ conatins infinitely many defined objects. ◊

Set-theoretic intuition tells us that there exists exactly one function from the empty set to an arbitrary set. That would suggest that for any set A there is an isomorphism between N_1 and $N_0 \to A$.
The next theorem shows that for intensional type theory this intuition is misleading in many respects.

Theorem 3.15

Assume that we interpret the singleton set N_1 in a non-extensional way by

$$(\{(0,0),(1,1)\},\{\{0\}\})$$

and the empty set N_0 also in a non-extensional way by

$$(\{(0,0)\},\{\})$$

then for any set A the sets N_1 and $N_0 \to A$ are not isomorphic.

For any set A the syntactically definable canonical map f from N_1 to $N_0 \to A$ sends all objects of type N_1 to the object $R_0\,([x{:}N_0]\,A)$ in $N_0 \to A$ which is realized by 0.
There is exactly one map g from $N_0 \to A$ to N_1 and this map is syntactically definable.
The composition $g \circ f$ is pointwise propositionally equal to identity (this can already be proved in the calculus as all objects of N_1 are provably propositionally equal).

If A contains more than one potential object then $f \circ g$ is not even pointwise propositionally equal to identity.
If A contains exactly one potential object then $f \circ g$ is the identity on $N_0 \to A$. Therefore the sets A and $N_0 \to A$ are weakly isomorphic by f and g.
Therefore under the current interpretation the sets N_1 and $N_0 \to N_0$ are weakly isomorphic.

But the sets N_1 and $N_0 \to N_0$ are not even weakly isomorphic if we interpret N_0 as the set $(\{(n,n) \mid n \in \omega\}, \{\})$.

Proof : As N_1 contains a potential object which is not actual whereas for any set A all objects of $N_0 \to A$ are actual it is impossible that N_1 and $N_0 \to A$ are isomorphic.
As all objects of $N_0 \to A$ are actual and N_1 contains exactly one actual oject there is a unique map g sending all objects of $N_0 \to A$ to the actual object of N_1. This map g is syntactically definable as $[f : N_0 \to A]\, c_0$ where c_0 is the constant denoting the unique actual object of N_1

Suppose that set A contains more than one potential object. Then the set $N_0 \to A$ contains more than one object and they are all actual. Now if $f \circ g$ were pointwise propositionally equal to identity then for all (necessarily actual) objects h of $N_0 \to A$ the set

$$\mathrm{Id}\,(N_0 \to A)\; h\; (f(g\,h))$$

would contain an actual object which would imply that h and $f(g\,h)$ are equal (as identity set contain an actual object only if their second and their third argument are really equal). But it is impossible that h and $f(g\,h)$ are really equal for all h in $N_0 \to A$ as f is a constant map and $N_0 \to A$ contains more than one actual object (due to the assumption that A contains more than one potential object).
If A contains exactly one potential object then $N_0 \to A$ contains exactly one object and this is actual. Therefore the map $f \circ g$ is equal to the identity map on $N_0 \to A$.
As one can (even syntactically) prove that $g \circ f$ is pointwise propositionally equal to identity the morphisms f and g constitute a weak isomorphism between the sets A and $N_0 \to A$.
Thus as in our interpretation the set N_0 contains exactly one potential object the sets N_1 and

$N_0 \to N_0$ are weakly isomorphic .

If N_0 is interpreted as the set $(\{ (n, n) \mid n \in \omega \}, \{\})$ then the set $N_0 \to N_0$ contains infinitely many actual objects whereas N_1 contains only one actual object. But then - as the function g is constant - the function $f \circ g$ is not pointwise propositionally equal to identy as this would entail that all objects h in $N_0 \to N_0$ were equal to the single object f(g h) . ◊

An interesting consequence of the last theorem is that in the model under consideration a certain proposition is wrong which has attracted some interest by type theorist recently.
In extensional type theory the inductive type of natural numbers N is isomorphic to the W-type $(Wx : N_2)\ A(x)$ where $A(c_0) = N_0$ and $A(c_1) = N_1$.
The isomorphism is given by a syntactically definable map f from N to $(Wx : N_2)\ A(x)$ mapping zero to $\sup(n_0, [x : N_0]\ R_0\ (Wx : N_2)\ A(x))$ and the syntactically definable map g from $(Wx : N_2)\ A(x)$ to N mapping any object of the form $\sup(n_0, f)$ to zero .
Now if f and g are interpreted in our model they would establish an isomorphism between N_1 and $N_0 \to (Wx : N_2)\ A(x)$ which cannot exist as $(Wx : N_2)\ A(x)$ contains more than one object.
More precisely, all objects of the form $\sup(n_0, h)$ with h in $N_0 \to (Wx : N_2)\ A(x)$ are defined and mapped to zero by g . Therefore g is not 1-1 even for actual objects and therefore the proposition stating that $f \circ g$ is pointwise propositionally equal to the identity function must be wrong.
Thus in our model a proposition which has been considered as unprovable in ICSTactually does not contain an actual object which means that there cannot be a proof term for this proposition. We do not know of any syntactical proof for the underivability of this proposition (as it practically seems to be almost impossible). Nevertheless a lot of people have been believing that this proposition is not formally provable in ICST.
We think that employing our model one can transform quite a lot of informal considerations on why a certain proposition is unprovable to a mathematical argument by showing that *in the model the proposition does not contain an actual object.*

3.7. Extensionality Principles for Functions Are Refuted in the Model

As a further illustration of this use of semantical methods we next will show that the extensionality principle for functions is refuted in our model and this even for number-theoretic functions.

Theorem 3.16

In the model where the type of sets is given by $\underline{\text{Set}}_1$ the proposition

$$\text{EXT} \equiv \{A : \text{Set}\}\{B : A \to \text{Set}\}\{f, g : \Pi\ A\ B\}$$
$$(\{x : A\}\ \text{Id}\ (B\ x)\ (\text{app}\ f\ x)\ (\text{app}\ g\ x)) \to \text{Id}\ (\Pi\ A\ B)\ f\ g$$

(expressing that if functions are pointwise propositionally equal then they are propositionally equal as functions) does not contain an actual object.

Proof : Suppose that the proposition EXT contains an actual object. We assume that Π is interpreted extensionally and therefore the type $\{x{:}A\}\ B(x)$ and the set $\Pi\ A\ B$ may be identified (as they are equal as mr-sets up to renaming of the elements of the underlying sets).
Take for A the set N_0 and for B the set N_1 constantly indexed over N_0.
We assume that N_0 is interpreted by the set $(\{(0,0)\}, \{\})$ and that N_1 is interpreted by the set $(\{(0,0), (1,1)\}, \{\{0\}\})$.
Then all objects of $\Pi\ A\ B$ are actual and there are exactly two of them which we call, say, f and g . Then - as N_0 does not contain any actual object - we know that all objects of the type $\{x : A\}\ \text{Id}\ (B\ x)\ (\text{app}\ f\ x)\ (\text{app}\ g\ x)$ are actual and there exists at least one (namely the one realized by 0) . As by assumption there is an actual object in EXT there exists also an actual object in the set $\text{Id}\ (\Pi\ A\ B)\ f\ g$. But due to our interpretation of identity type if $\text{Id}\ (\Pi\ A\ B)\ f\ g$ contains a total object then f and g must be equal contradicting the assumption that they are different. ◊

Next we will show that even if A contains only actual objects the corresponding instance of EXT is not be valid in the model.
More precisely, we will show that for functions between natural numbers the extensionality principle does not hold.

Theorem 3.17

In the model where the type of sets is given by $\underline{\text{Set}}_1$ and the type N of natural numbers is interpreted by $(\{(n, n) \mid n \in \omega\}, \{\{n\} \mid n \in \omega\})$ the proposition

$$\text{EXT-N} \equiv \{f, g : N \to N\}\ (\{x : N\}\ \text{Id}\ N\ (\text{app}\ f\ x)\ (\text{app}\ g\ x)) \to \text{Id}\ (N \to N)\ f\ g$$

does not contain any actual object.

Proof : We will show that the assumption that the proposition EXT-N is inhabited by an actual object entails that it would be decidable whether total recursive functions are equal. But this problem is known to be undecidable due to the Kreisel-Lacombe-Shoenfield-Theorem, see e.g. [Ro].

For any total recursive functions f and g let h(f,g) in {x : N} Id N (app f x) (app g x) be defined as follows :

if $f(n) = g(n)$ then $h(f,g)(n) = \{1\}$

if $f(n) \neq g(n)$ then $h(f,g)(n) = \{0\}$

Obviously, a realizer for h can be computed from codes for f and g.
It is clear from the definition of h that the object h(f,g) in {x : N} Id N (app f x) (app g x) is actual if and only if f and g are equal.

Now if we assume that EXT-N contains an actual object p then the function

$$e \equiv [f, g : N \to N]\ p\, f\, g\, (h(f,g))$$

has the following property : if $f = g$ then h(f,g) is actual (as all objects in $N \to N$ are actual) and therefore e f g is an actual object in Id $(N \to N)$ f g which therefore is realized by 1 and if $f \neq g$ then the object e f g in Id $(N \to N)$ f g must be realized by 0 as due to the assumption $f \neq g$ the only object in Id $(N \to N)$ f g is {0} .

Therefore any realizer for e provides a Gödel number for a decision procedure for the extensional equality of total recursive functions. ◊

3.8. On the Relation Between Identity Sets and Leibniz Equality

Now we turn our our attention to the study of the relations between Martin-Löf's identity sets and Leibniz equality. We will establish some positive and some negative results w.r.t. to derivability in ICST. Just to remember the Leibniz equality on a type A is defined as the predicate

$$\mathrm{Eq}\, A \equiv [x,y : A]\ \{P : A \to \mathrm{Set}\}\ (P\, x) \to (P\, y) \in \{x, y : A\}\ \mathrm{Set}$$

where we assume that Set is *closed under impredicative unversal quantification*, i.e.that the product of a family of sets indexed over an arbitary set again is (isomorphic to) a set. Thus Leibniz equality types are always in Set .

We now will discuss problems of *equivalence* and *isomorphism between identity types and Leibniz equality types.*

Of course, one always can prove that Id A x y implies Eq A x y using the J-eliminator as shown by the following Lemma.

Lemma 3.18

The term

$$
\begin{aligned}
f \;\equiv\; [A{:}Set]\;\; J\;\; A \\
&([x\,,\,y : A][z : Id\;A\;x\;y]\;Eq\;A\;x\;y) \\
&([x : A]\;[P : A \to Set]\;[p : P\;x]\;p)
\end{aligned}
$$

is of type

$$\{A : Set\}\;\{x\,,\,y : A\}\;(Id\;A\;x\;y) \to Eq\;A\;x\;y \qquad \Diamond$$

Next we will discuss the more problematic converse direction.

In the LEGO system, see e.g. [LP], if one defines identity types on the level of Type(0) then it is impossible to prove that Eq A x y implies Id A x y because one may interpret the type Prop trivially (that is where Leibniz equality types live !) and identity sets in an extensional way. By a trivial interpretation of Prop we mean that there exists only one object in Prop and this unique propositional type contains exactly one object. Then if one chooses $x \neq y$ then Eq A x y is inhabited but Id A x y is not and therefore there cannot exist a map from the type Eq A x y to the type Id A x y .
There also exists a nontrivial model for this situation. Interpret Prop by $\underline{Set}_1$ and Type(0) by $\underline{Set}$ (see Def. 3.3) and interpret identity types extensionally. Then again for distinct objects x and y of type A there cannot exist a morphism from Eq A x y to Id A x y as Eq A x y contains at least one potential object (e.g. the one realized by 0) and the empty type Id A x y does not even contain a potential object.

The situation is quite different if data types and therefore also identity types live on the level of Prop which from now on again will be called Set .
Then one can prove the equivalence of both notions of equality in the following way.

Lemma 3.19

The term

$$g \;\equiv\; [A : Set]\;[x\,,\,y : A]\;[p : Eq\;A\;x\;y]\;p\;(Id\;A\;x)\;(r\;A\;x)$$

is of type

$$\{A : \mathrm{Set}\}\ \{x\,,\, y : A\}\ (\mathrm{Eq}\ A\ x\ y) \rightarrow \mathrm{Id}\ A\ x\ y$$

It holds that

$$A : \mathrm{Set}\,,\, x : A \vdash g\ A\ x\ x\ (f\ A\ x\ x\ (r\ A\ x)) = r\ A\ x \in \mathrm{Id}\ A\ x\ x$$

and therefore the term

$$\begin{array}{l} [A : \mathrm{Set}]\ \ J\ A\ \ ([x\,,\, y : A][z : \mathrm{Id}\ A\ x\ y]\ \mathrm{Id}\ (\mathrm{Id}\ A\ x\ y)\ z\ (g\ A\ x\ y\ (f\ A\ x\ y\ z))) \\ \qquad\qquad\quad ([x : A]\ r\ (\mathrm{Id}\ A\ x\ x)\ (r\ A\ x)) \\ \in \\ \{A : \mathrm{Set}\}\ \{x\,,\, y : A\}\ \{z : \mathrm{Id}\ A\ x\ y\}\ \mathrm{Id}\ (\mathrm{Id}\ A\ x\ y)\ z\ (g\ A\ x\ y\ (f\ A\ x\ y\ z)) \end{array}$$

i.e. for any set A and objects x and y in A for the functions

$$f\ A\ x\ y \in \mathrm{Id}\ A\ x\ y \rightarrow \mathrm{Eq}\ A\ x\ y \qquad g\ A\ x\ y \in \mathrm{Eq}\ A\ x\ y \rightarrow \mathrm{Id}\ A\ x\ y$$

it holds that $(g\ A\ x\ y) \circ (f\ A\ x\ y)$ is pointwise propositionally equal to identity. Thus in a weak sense identity types appear as retracts of Leibniz equality types.

Proof : Straightforward, but tedious, by applying the conversion rule. $\diamond$
Of course, then there immediately arises the question whether $(g\ A\ x\ y) \circ (f\ A\ x\ y)$ might convert to the identity function on $\mathrm{Id}\ A\ x\ y$. This question will be answered negatively by semantical methods.
Furthermore we will show that $(f\ A\ x\ y) \circ (g\ A\ x\ y)$ is not even pointwise propositionally equal to the identity on $\mathrm{Eq}\ A\ x\ y$ and that Eq cannot be endowed with an eliminator J ..

But before this we have to look more carefull what Leibniz equality types look like when interpreted in the model where Set is interpreted by $\underline{\mathrm{Set}}_1$.

Theorem 3.20

In the model where Set is interpreted by $\underline{\mathrm{Set}}_1$ Leibniz equality types can be characterized in the following way.

For any set A and a in A the type $\mathrm{Eq}\ A\ a\ a$ is canonically isomorphic to the set $(R\,,\, P)$ where

$n\ R\ m$ iff both n and m are codes for the same total recursive function

and
for all natural numbers k it holds that $\{n\}(k) = k$ or $\{n\}(k) = 0$

$P = \{ \{ n \mid \{n\}(k) = k \text{ for all } k \in \omega \}\}$

For any set A and a, b in A with $a \neq b$ the type $\mathrm{Eq}\ A\ a\ b$ is canonically isomorphic to the set $(\{ (n, m) \mid \{n\} = \{0\} = \{m\} \}, \{ \})$.

The map $f\ A\ a\ b$ maps any non-actual object to the object realized by 0 and it maps the unique actual object - if it exists - to the equivalence class consisting of all codes for the identity function on natural numbers.

The map $g\ A\ a\ b$ maps the equivalence class $\{n\}$ to the object realized by $\{n\}(1)$.

Proof : A realizable (potential) object of type $\mathrm{Eq}\ A\ a\ b = \{P : A \rightarrow \mathrm{Set}\}\ (P\ a) \rightarrow (P\ b)$ must necessarily be uniform as $A \rightarrow \mathrm{Set}$ carries a trivial realizablity structure.

Therefore a potential object of $\mathrm{Eq}\ A\ a\ a$ is - up to evident isomorphism - determined by a partial recursive function which for any $R \in$ PER-0 realizes an endofunction on R. Clearly such an $\{n\}$ maps any natural number k either to itself or to 0 (if $\{n\}(k) \neq k$ and $\{n\}(k) \neq 0$ then it does not realize an endofunction on a per having besides 0 only k in its carrier) and any such n clearly realizes an endofunction for any any $R \in$ PER-0.
Only codes for the total identity function on natural numbers can realize an actual endofunction for all sets as for any different natural numbers k and l one always can find a set where k realizes an actual object and l realizes a non-actual object.

If a and b are different objects in A then an object of $\mathrm{Eq}\ A\ a\ a$ is - up to evident isomorphism - determined by a partial recursive function which for all $R, S \in$ PER-0 realizes a function from R to S. Clearly any code for the constant function with value 0 realizes a function from R to S for all $R, S \in$ PER-0. Any natural number n realizing a function from R to S for all $R, S \in$ PER-0 clearly must code the constant function with value 0 (choose for $R = \omega \times \omega$ and $S = \{ (0,0) \}$).
Of course, the constant function with value 0 does not always realize an actual object because one may take (R, P) and (S, Q) where $[0]_S$ is not in Q and P is non-empty.

Due to the definition of the eliminator J the mapping $f\ A\ a\ b$ maps any non-actual object of the set $\mathrm{Id}\ A\ a\ b$ to the object in $\mathrm{Eq}\ A\ a\ b$ realized by 0.
Due to the definition of f the function $f\ A\ a\ a$ maps the actual object of type $\mathrm{Id}\ A\ a\ a$ to the actual object of $\mathrm{Eq}\ A\ a\ a$ realized by a code for the total idenity function on natural numbers.

By definition the map $g\ A\ a\ b$ applies an object of type $\mathrm{Eq}\ A\ a\ b$ (after applying it to the family Id A a) to the canonical object r A a of type Id A a a . As the object r A a is realized by 1 an object in (the canonically isomorphic copy) of Eq A a b realized by n is mapped by g A a b to the object realized by {n}(1) . ◊

Theorem 3.21

For any set A and object a in A the composition (g A a a) ∘ (f A a a) is not equal to the identity on Id A a a if we interpret identity sets in a way such that idenity sets *always contain a non-actual object different from the one realized by* 0 .
Therefore (g A a a) ∘ (f A a a) cannot be convertible to the identity function.

For any set A and object a in A the set corresponding to the proposition stating that the composition (f A a a) ∘ (g A a a) is pointwise propositionally equal to identity does not contain an actual object.
Therefore this proposition cannot be proved in the calculus.

Proof : Any non-actual object of type Id A a a not realized by 0 is mapped by f A a a to the object in Eq A a a realized by 0 and therefore (according to Theorem 3.20) this object is mapped by g A a a to the object of Id A a a realized by 0 (as 0 codes the function with constant value 0). Thus (g A a a) ∘ (f A a a) is different from the identity on Id A a a .

We now consider the interpretation where identity sets contain exactly one non-actual object and this object is realized by 0 .
Suppose that there were an actual object in the set corresponding to the proposition that the composition (f A a a) ∘ (g A a a) is pointwise propositionally equal to identity.
Then - due to the interpretation of identity sets - a realizer for this proof object would provide a partial recursive function h which terminates for all arguments in the set

$$D = \{\, n \in \omega \mid \{n\}(k) = k \text{ or } \{n\}(k) = 0 \text{ for all } k \in \omega \,\}$$

and whose behaviour for $n \in D$ is specified by the following case analysis :

$h(n) = 1$	if n is code for the total identity function
$h(n) \in \{0,1\}$	if n is a code for the total function with constant value 0
$h(n) = 0$	otherwise

But the existence of such a partial recursive function h would make the halting problem decidable as shown by the following argument.

Let s be a total recursive function which for any natural number n computes a code of the total recursive function t(n) defined as follows :

$$t(n)(m) = m \qquad \text{if } m = 1 \text{ or } T(n, n, k) \text{ is false for all } k < m$$

$$t(n)(m) = 0 \qquad \text{otherwise}$$

(where T is of course Kleene's T-predicate).
Then t(n) is the total identity function if T(n,n,k) is false for all k , i.e. the computation of {n}(n) diverges, and t(n) is different from the identity function *and* from the total function with constant value 0 if the computation of {n}(n) terminates. But then

$$h(s(n)) = 1 \qquad \text{if } \{n\}(n) \text{ does not terminate}$$

$$h(s(n)) = 0 \qquad \text{if } \{n\}(n) \text{ terminates .}$$

As this would make the halting problem decidable there cannot exits an actual proof object for the proposition expressing that (f A a a) ∘ (g A a a) is pointwise propositionally equal to identity ◊

Thus we can conclude that the natural candidates f and g for establishing an isomorphism between identity sets and Leibniz equality sets don't do this job as (f A a a) ∘ (g A a a) is not even pointwise propositionally equal to the identity on Eq A a a .

We will finish our comparison of Martin-Löf's identity sets and Leibniz equality sets by showing that in the model where Set is interpreted by $\underline{\text{Set}}_1$ such an isomorphism never can exist as it is impossible to endow Leibniz equality types with the structure of identity types.
The proof of this theorem will contain a rather sharp characterization of how interpretations of identity types may look like in this model.

Theorem 3.22

For any interpretation of identity sets in the model where Set is interpreted by $\underline{\text{Set}}_1$ the following conditions are satisfied for all sets A and actual objects a , b in A :

(a) if Id A a b contains an actual object then a = b

(b) there can be at most one actual object in Id A a b

(c) the realizers of the unique actual object r A a in Id A a a
can be separated effectively from the realizers for the non-actual objects

Furthermore, Leibniz equality sets cannot be endowed with the structure required for identity sets as for all sets A and actual objects a in A the realizers for the unique actual object in the Leibniz equality set Eq A a a cannot be separated effectively from the realizers for the non-actual objects.

Proof : Let A be a set and C a family in {x , y : A} {z : Id A x y} Set such that for all potential objects x , y in A and z in Id A x y : $C\ x\ y\ z = N_1$ if $x = y$ and $z = r\ A\ x$ and $C\ x\ y\ z = N_0$ otherwise.
Then $d \equiv [x : A]\ c_0$ is an actual object in {x : A} C x x (r A x) and J A C is an actual object in {x , y : A} {z : Id A x y} C x y z .
As J A C is actual it maps actual objects to actual objects.

Let a and b be actual objects in A .

Now if there were an actual object c in Id A a b then J A C d a b c were an actual object in C a b c . But as N_0 does not contain an actual object it follows that $a = b$.

If there were an actual object c in Id A a a with $c \neq r\ A\ a$ then J A C d a a c were an actual object in N_0 . But as this is impossible there cannot be an actual object in Id A a a different from r A a .

The actual object J A C d a a maps the actual object r A a to the actual object c_0 realized by 1 and all other objects in Id A a a to the non-actual object in N_0 which is realized by 0 . Therefore a realizer for J A C d a a maps realizers for r A a to 1 and realizers for other objects to 0 .
Thus these two disjoint sets of realizers can be separated effectively.

From the proof of Theorem 3.20 it follows immediately that the realizers of the actual object in Eq A a a cannot be separated effectively from the realizers of the non-actual objects in the set Eq A a a as this would give rise to a decision procedure for the halting problem.

Therefore it is impossible to endow Eq with the structure required for identity sets because for any set A containing an actual object a the interpretation of the actual object r A a must be equal to the unique actual object of type Eq A a a whose realizers cannot be separated effectively from the realizers for the other non-actual objects. ◊

Remark. Notice that the theorem above does not say anything about the definedness of the object r A a if a is a non-actual object of type A .

In fact one can modify any interpretation of identity sets in the model where Set is interpreted by $\underline{\mathrm{Set}}_1$ to another correct interpretation simply by eliminating for any non-actual object a in a set A the object r A a from the actual objects of set Id A a a but still keeping it as a potential object.

We conclude our investigations of the model where Set is interpreted by $\underline{\mathrm{Set}}_1$ by the following theorem.

Theorem 3 .23

For any interpretation of identity sets in the model where Set is interpreted by $\underline{\mathrm{Set}}_1$ it holds that for any set A , family C in {x , y : A} {z : Id A x y} Set , potential objects a , b in A and for any object c in Id A a b different from r A a the object J A C d a b c in C a b c is realized by 0 .

Proof : The types Set and {x , y : A} {z : Id A x y} Set carry a trivial realizability structure and therefore the eliminator J is uniform in A and C . Suppose that n is a realizer for J and let n_1 , n_2 , n_3 , n_4 , n_5 , n_6 be realizers for A , C , d , a , b , c , respectively, then n applied to n_1 - n_6 realizes the object J A C d a b c . Assume that c is different from the object r A a . Then consider the family D in {x , y : A} {z : Id A x y} Set such that for all potential objects x , y in A and z in Id A x y : D x y z = C x y z if x = y and z = r A x and D x y z = N_0 otherwise. As the number n_2 realizes also D the result of n applied to n_1 - n_6 realizes the object J A D d a b c in N_0 which is realized only by 0 . ◊

This defect can be avoided by working relative to a new Gödel numbering ϕ of partial recursive functions where for all $n, m \in \omega$: $\phi(2n)(m) = 2n$ and $\phi(2n+1)(m) \cong \{n\}(m)$ and interpreting the type Set by $\underline{\mathrm{Set}}(\{ R \mid R$ is a per with $2n\ R\ 2n$ for all $n \in \omega \})$. Then for any even number n one can interpret J as follows : J A C d a b c = d a if a = b and c = r A a and otherwise J A C d a b c is the unique object in C a b c realized by n . Similarly one can interpret K . Of course, for different even numbers n this gives rise to different interpretations of J and K .

3.9. A Model Based on Effective Domains

Next we will discuss models where the ambient logical framework still is interpreted in the category **mr-Set** but the type Set will be interpreted as the mr-set with trivial realizability structure whose potential objects are effective domains together with a distinguished subset of

actual objects and where all objects are actual.
Using a result from [Str2] we will show that this provides a model of where *it is impossible to interpret strong sum types*. The impossibility of interpreting strong sum types hinges on the fact that *the sum of a family of effective domains indexed over an effective domain need not be an effective domain again.*

Let us quickly review the notion of *complete extensional per with bottom* as originally introduced in [FMRS].
An extensional per R is characterized by a set D of natural numbers together with a collection $\mathfrak{D}$ of subsets of D such that for any $A \in \mathfrak{D}$ there exists a natural number n such that for all $m \in D$: $\{n\}(m) = 0$ if $m \in A$ and $\{n\}(m)$ diverges if $m \notin A$ (the collection of all such numbers n is denoted as $\mathrm{real}(D,A)$). Given such D and $\mathfrak{D}$ the associated per $\mathrm{exper}(D,\mathfrak{D})$ is described as follows : $n \; \mathrm{exper}(D,\mathfrak{D}) \; m$ iff for some $A \in \mathfrak{D}$ both n and m are contained in $\mathrm{real}(D,A)$. It is easy to see that for any extensional per R there exist unique D and $\mathfrak{D}$ such that $R = \mathrm{exper}(D,\mathfrak{D})$.
A *complete extensional per* is an extensional per closed under suprema of effective increasing chains. More precisely, an extensional per $R = \mathrm{exper}(D,\mathfrak{D})$ is closed under suprema of effective chains iff for any total recursive function f such that for all $n \in \omega$: $f(n) \, R \, f(n)$ and for all $m > n$ and $k \in D$ if $\{f(n)\}(k) = 0$ then $\{f(m)\}(k) = 0$ it holds that $\mathrm{real}(D,A) \in \omega/R$ where $A = \{ \ k \mid \{f(n)\}(k) = 0 \ \text{ for some } n \in \omega \ \}$.

A *complete extensional per with bottom* is a complete extensional per containing all codes for the totally undefined partial recursive function in its carrier.
In [FMRS] it has been shown that complete extensional pers with bottom form an internally complete full subcategory of r-**Set** .

Theorem 3.24

Let $\mathrm{CExPer}_\perp$ be the collection of all complete extensional pers with bottom. If one interprets the type Set by <u>Set</u>($\mathrm{CExPer}_\perp$) then this gives a model of impredicative universal quantification, intensional identity types but it is impossible to interpret intensional strong sums.

Proof : It has been shown in [FMRS] and in [Str2] that $\mathrm{CExPer}_\perp$ is closed under arbitrary products internal to r-**Set** . Products internal to mr-**Set** are computed as in r-**Set** and actual objects are characterized as those realizable functions which preserve actual objects.

Intensional identity types can be interpreted as follows.

Let i_0 be a code for the empty function which nowhere terminates and let i_1 be a code for the constant function with value 0 .
We further put $I_0 = \{ n \mid \{n\} = \{i_0\}\}$ and $I_1 = \{ n \mid \{n\} = \{i1\} \}$.

For any set A and objects a , b in A we define the set Id A a b according to the following case analysis :

- if $a = b$ then

$$\text{Id A a b} = ((I_0 \times I_0) \cup (I_1 \times I_1) , \{I_0\})$$

- if $a \neq b$ then

$$\text{Id A a b} = (I_0 \times I_0 , \{ \})$$

Thus, intutively, if $a = b$ then the corresponding identity set contains the actual object I_1 and the non-actual, only potential object I_0 and if $a \neq b$ then the corresponding identity set contains only the potential object I_0 .

The constructor r is interpreted as follows. For any set A and object a in A

$$\text{r A a} = I_1 \in \text{Id A a a}$$

and, obviously, the function r is realizable and it preserves actual objectts as the object $I_1 \in$ Id A a a is always actual.

The eliminator J is defined as follows. Given a set A , a family C of sets indexed over x , y $\in$ A and $z \in$ Id A x y and apotential object $d \in \{x{:}A\}$ C x x (r A x) then J A C d is defined according to the following case analysis :

- for x in A

$$\text{J A C d x x } I_1 = \text{d x}$$

- for x , y in A

$$\text{J A C d x y } I_0 = \text{the unique object of C x y } I_0 \text{ realized by } i_0$$

The morphism J A C d is realized by an appropriate currying of the following algorithm : let e be a realizer for d then given realizers n , m , k for x , y , z , respectively, and an arbitrary natural number l then a(n,m,k,l) = {{e}(x)}(l) if {k}(0) terminates and undefined otherwise.

Similarly one can interpret the eliminator K .

In order to prove that one cannot interpret intensional strong sums in $\underline{\text{Set}}(\text{CExPer}_\perp)$ we have to explain the notion of Σ-separability, see [Str2].

Let Σ be exper({0},{{},{0}}) which is a complete extensional per with bottom providing an effective version of the Scott domain with two objects (sometimes called the "schizophrenic object").
We say that a per R is Σ-*separable* iff for any two different objects x and y in (the r-set represented by) R there is a realizable $p : R \to \Sigma$ such that $p(x) \neq p(y)$. Clearly, any extensional per is Σ-separable.
By declaring all objects of Σ as actual objects we can consider Σ as an object in the category **mr-Set** and the morphisms from an mr-set $\underline{X}$ to Σ are exactly the realizable morphisms between their underlying r-sets. Then the notion of Σ-separablity can be defined for the category **mr-Set** in an analogous way. Therefore an mr-set is Σ-separable in the category **mr-Set** iff its underlying r-set is Σ-separable in the category **r-Set** .

Now suppose we had an interpretation of intensional strong sums, i.e. an interpretation of Σ , pair and E satisfying the corresponding judgemental equalities.
Then for any set A and any family $B : A \to \text{Set}$ the object pair A B would be an injective realizable function from the context x : A , y : B(x) to the set Σ A B . The injectivity of the function pair A B follows from the fact that even in the sense of judgemental equality it holds that $\text{pr}_0(\text{pair A B a b}) = a \in A$ and $\text{pr}_1(\text{pair A B a b}) = b \in B(\text{pr}_0(\text{pair A B a b}))$.
As shown in [Str2] (Theorem 2 (1)) there is a special choice for A and B such that the context x : A , y : B(x) is not Σ-separable whereas the set Σ A B is Σ-separable (as its underlying per is extensional). But as pair A B is an injective realizable function from the context x : A , y : B(x) to the set Σ A B it follows that x : A , y : B(x) is also Σ-separable in contradiction to our choice of A and B .

Thus strong sums cannot be interpreted in this model. ◊

Another model refuting the interpretation of intensional strong sums is provided by interpreting the type Set by $\underline{\text{Set}}(\{ R \to \Delta(\omega) \mid R \in \text{PER}\})$ where $\Delta(\omega) = \{ (n, n) \mid n \in \omega \}$.
In [Str3] it has been shown that $\{ R \to \Delta(\omega) \mid R \in \text{PER}\}$ gives rise to a full subcategory of **r-Set** which is internally closed under arbitrary products but not closed under small strong sums. Therefore the same argument as for $\underline{\text{Set}}(\text{CExPer}_\perp)$ can be performed in order to show that strong sums cannot be interpreted in the model where Set is interpreted by the mr-set

$\underline{\text{Set}}(\{ R \to \Delta(\omega) \mid R \in \text{PER}\})$.
This latter interpretation allows to interpret also (extensional) natural numbers as the type associated with the set $(R_N , \omega/R_N)$ with $R_N := \{ (0 , 0) \} \to \Delta(\omega)$ is a natural number object in **mr-Set** .
This seems to be an advantage of this particular model as not even for the intensional version of the natural number type it is clear how it can be interpreted in $\underline{\text{Set}}(\text{CExPer}_\perp)$.

3.10 Conclusion

In this last chapter we have demonstrated that by a slight modification of the well-known realizability model for extensional Constructive Set Theory it is possible to define *mathematical, non-syntactical models* of ICST which are *fully intensional.* We have shown that these models are suitable for refuting most of the sequents which are derivable in extensional Constructive Set Theory but not in ICST.
More generally we suggest the following informal method for checking the appropriateness of some proposed extension of ICST. First try to interpret it in the fully intensional mathematical model(s) studied in this chapter. If this is easy - and experience tells that if the extension is sound then the interpretation is straighforward - then make a more refined syntactic analysis checking whether the added conversion rules do not violate the required meta-mathematical properties as confluence and strong normalization. But as these syntactic checks can be considerably hard and therefore time comsuming it is better to first do the "semantic check" which is easier and faster and allows to rule out wrong suggestions.
An interesting such application would be e.g. to interpret the extension of ICST suggested by Th. Coquand in [Coq] .

Appendix 1

The Logical Framework

We first give a presentation of the so called Logical Framework where (higher order) rules can be expressed. It is a dependently typed λ-calculus which also provides the corner stone of the various languages of the AUTOMATH family developed by N. deBruijn starting in the 60s.

To start with we give the raw syntax of the logical framework.

We assume an a priori given syntactic category Var of *object variables* and based on it we define the syntactic categories TyExp of *raw type expressions* and ObExp of *raw object expressions* in a BNF-like manner :

TyExp ::= {Var : TyExp} TyExp

ObExp ::= Var | [Var : TyExp] ObExp | ObExp(ObExp)

We use A , B , C possibly decorated with primes or indices as meta variables ranging over objects of the syntactic category TyExp and t , s possibly decorated with primes or indices as meta variables ranging over objects of the syntactic category ObExp .

Raw type expressions of the form {x : A} B are called *raw product type expressions*, raw object expressions of the form x are called *raw variables*, raw object expressions of the form [x : E] t are called *raw functional abstractions* and raw object expressions of the form t (s) are called raw *function applications.*

A raw context is a syntactic expression of the form

$$x_1 : A_1 , \ldots , x_i : A_i , \ldots , x_n : A_n$$

whose intended meaning is that of a declaration of variables. Of course, proper contexts will satisfy the condition that for any i with $1 \leq i \leq n$ the free variables of A_i are contained in the list $x_1 , \ldots , x_{i-1}$. We use capital Greek letter Γ, Δ ... to range over raw contexts.

In type theory there are four different forms of so called judgements (germ.*Urteil*) :

A	A is a well formed type
$A = B$	A and B are well formed and equal types
$t \in A$	t is a well formed object of the well formed type A
$t = s \in A$	t and s are well formed and equal objects of the well formed type A

These forms of judgements will be ranged over by the metavariable $\mathcal{J}$.

A *raw sequent* is a syntactic expression of the form $\Gamma \vdash$ or of the form $\Gamma \vdash \mathcal{J}$ with the following intuitive meanings

$\Gamma \vdash$	Γ is a well formed context
$\Gamma \vdash \mathcal{J}$	the judgement $\mathcal{J}$ is valid w.r.t. the well formed context Γ

Now we are going to state the rules of our dependently typed λ-calculus called **LF** (Logical Framework). They inductively define the collection of all formally provable sequents.

Context Formation Rules

EMPTY

$$\frac{}{\vdash}$$

CONT-INTRO

$$\frac{\Gamma, \Delta \vdash \quad \Gamma \vdash A}{\Gamma, x : A , \Delta \vdash} \quad (\text{ if } x \notin \mathrm{Var}(\Gamma) \cup \mathrm{Var}(\Delta))$$

CONT-SUB

$$\frac{\Gamma, x : A , \Delta \vdash \quad \Gamma \vdash t \in A}{\Gamma , \Delta [t/x] \vdash}$$

Structural Rules

ASS

$$\frac{\Gamma, x : A , \Delta \vdash}{\Gamma, x : A , \Delta \vdash x \in A}$$

THIN

$$\frac{\Gamma, \Delta \vdash \mathcal{J} \qquad \Gamma \vdash A}{\Gamma, x : A , \Delta \vdash \mathcal{J}} \; (x \notin \mathrm{Var}(\Gamma) \cup \mathrm{Var}(\Delta))$$

SUB

$$\frac{\Gamma, x : A , \Delta \vdash \mathcal{J} \qquad \Gamma \vdash t \in A}{\Gamma, \Delta[t/x] \vdash \mathcal{J}[t/x]}$$

Equality Rules

REFL-TYP

$$\frac{\Gamma \vdash A}{\Gamma \vdash A = A}$$

REFL-OBJ

$$\frac{\Gamma \vdash t \in A}{\Gamma \vdash t = t \in A}$$

SYMM-TYP

$$\frac{\Gamma \vdash A = B}{\Gamma \vdash B = A}$$

SYMM-OBJ

$$\frac{\Gamma \vdash t = s \in A}{\Gamma \vdash t = s \in A}$$

TRANS-TYP

$$\frac{\Gamma \vdash A = B \quad \Gamma \vdash B = C}{\Gamma \vdash A = C}$$

TRANS-OBJ

$$\frac{\Gamma \vdash t_1 = t_2 \in A \qquad \Gamma \vdash t_2 = t_3 \in A}{\Gamma \vdash t_1 = t_3 \in A}$$

REPL1

$$\frac{\Gamma \vdash t = s \in A \qquad \Gamma, x : A, \Delta \vdash B}{\Gamma, \Delta[t/x] \vdash B[t/x] = B[s/x]}$$

REPL2

$$\frac{\Gamma \vdash t = t' \in A \qquad \Gamma, x : A, \Delta \vdash s \in B}{\Gamma, \Delta[t/x] \vdash s[t/x] = s[t'/x] \in B[t/x]}$$

CONV 1

$$\frac{\Gamma \vdash A = B \qquad \Gamma \vdash t \in A}{\Gamma \vdash t \in B}$$

CONV 2

$$\frac{\Gamma \vdash A = B \qquad \Gamma \vdash t = s \in A}{\Gamma \vdash t = s \in B}$$

Rules for Products of Types

PROD-FORM

$$\frac{\Gamma, x : A \vdash B}{\Gamma \vdash \{x : A\}\, B}$$

PROD-EQU

$$\frac{\Gamma \vdash A_1 = A_2 \qquad \Gamma, x : A_1 \vdash B_1 = B_2}{\Gamma \vdash \{x : A_1\}\, B_2 = \{x : A_2\}\, B_2}$$

PROD-INTRO

$$\frac{\Gamma, x : A \vdash t \in B}{\Gamma \vdash [x : A]\, t \in \{x : A\}\, B}$$

PROD-INTRO-EQU

$$\frac{\Gamma, x : A \vdash t = s \in B}{\Gamma \vdash [x : A]\, t = [x : A]\, s \in \{x : A\}\, B}$$

PROD-ELIM

$$\frac{\Gamma, x : A \vdash B \qquad \Gamma \vdash t \in \{x : A\}\, B \qquad \Gamma \vdash s \in A}{\Gamma \vdash t\,(s) \in B[s/x]}$$

PROD-ELIM-EQU

$$\frac{\Gamma, x : A \ \vdash B \quad \Gamma \vdash t = t' \in \{x : A\}\, B \quad \Gamma \vdash s = s' \in A}{\Gamma \vdash t\,(s) = t'(s') \in B[s/x]}$$

β-RULE

$$\frac{\Gamma, x : A \ \vdash B \quad \Gamma, x : A \ \vdash t \in B \quad \Gamma \ \vdash s \in A}{\Gamma \ \vdash ([x : A]\, t)\,(s) = t[s/x] \in B[s/x]}$$

η-RULE

$$\frac{\Gamma, x : A \vdash B \quad \Gamma \vdash t \in \{x : A\}\, B}{\Gamma \vdash [x : A]\ t\,(x) = t \in \{x : A\}\, B}$$

Appendix 2

Constructive Set Theory in the Logical Framework

Martin-Löf's CST (Constructive Set Theory) can be formulated inside LF by assuming a type Set of sets. But we also want to consider objects A of type Set as types which is done by postulating a family El of types indexed over Set which associates with any $A \in \mathrm{Set}$ its type El(A) of elements of set A .

Sets as Types

$\vdash \mathrm{Set}$ $\qquad\qquad$ $A : \mathrm{Set} \vdash \mathrm{El}(A)$

The other sets of CST are defined by constants for *set formation*, constants for *constructing canonical elements* and constants for *elimination operations* which allow to define functions from a prescription of their behaviour on canonical elements. Furthermore conversion rules are stated which describe how functions obtained by elimination operations evaluate when given canonical objects as arguments.

Products of Families of Sets

$\Pi \in \{A : \mathrm{Set}\}\{B : \{x : \mathrm{El}(A)\}\ \mathrm{Set}\ \}\ \mathrm{Set}$

$\mathrm{fun} \in \{A : \mathrm{Set}\}\{B : \{x : \mathrm{El}(A)\}\ \mathrm{Set}\}\{f : \{x : \mathrm{El}(A)\}\ \mathrm{El}(B(x))\}\ \ \mathrm{El}(\Pi(A)(B))$

$\mathrm{F} \in \{A : \mathrm{Set}\}\{B : \{x : \mathrm{El}(A)\}\ \mathrm{Set}\}\{C : \{z : \mathrm{El}(\Pi(A)(B))\}\ \mathrm{Set}\}$
$\quad \{d : \{f : \{x : \mathrm{El}(A)\}\ \mathrm{El}(B(x))\}\ \ \mathrm{El}(C(\lambda(A)(B)(f)))\ \}\ \{c : \mathrm{El}(\Pi(A)(B))\}$
$\qquad \mathrm{El}(C(c))$

together with the conversion rule

$A : \mathrm{Set},\ B : \{x : \mathrm{El}(A)\}\ \mathrm{Set},\ C : \{z : \mathrm{El}(\Pi(A)(B))\}\ \mathrm{Set},$
$d : \{f : \{x : \mathrm{El}(A)\}\ \mathrm{El}(B(x))\}\ \mathrm{El}(C(\lambda(A)(B)(f))),\ f : \{x : \mathrm{El}(A)\}\ \mathrm{El}(B(x))$
$\vdash$
$F(A)(B)(C)(d)(\lambda(A)(B)(f)) = d(f) \in \mathrm{El}(C(\lambda(A)(B)(f)))$

A fairly useful defined notion is *function application* which in terms of F can be defined as follows

$\mathrm{apply} \equiv [A : \mathrm{Set}]\ [B : \{x : \mathrm{El}(A)\}\ \mathrm{Set}\ ]\ [c : \mathrm{El}(\Pi(A)(B))]\ [a : \mathrm{El}(A)]$
$\quad F\ A\ B\ ([z: \mathrm{El}(\Pi(A)(B))]\ B(a))\ ([f : \{x : \mathrm{El}(A)\}\ \mathrm{El}(B(x))]\ f(a))\ c$

for which one can derive

$\vdash \mathrm{apply} \in \{A : \mathrm{Set}\}\ \{B : \{x : \mathrm{El}(A)\}\ \mathrm{Set}\}\ \{c : \mathrm{El}(\Pi(A)(B))\}\ \{a : \mathrm{El}(A)\}\ \mathrm{El}(B(a))$

and the following judgemental equality describing how to evaluate application terms

$A : \mathrm{Set},\ B : \{x : \mathrm{El}(A)\}\mathrm{Set},\ f : \{x : \mathrm{El}(A)\}\ \mathrm{El}(B(x)),\ a : \mathrm{El}(A)$
$\vdash$
$\mathrm{apply}(A)(B)(\lambda(A)(B)(f))(a) = f(a) \in \mathrm{El}(B(a))$

Sums of Families of Sets

$\Sigma \in \{A : \mathrm{Set}\}\ \{B : \{x : \mathrm{El}(A)\}\ \mathrm{Set}\}\ \mathrm{Set}$

$\mathrm{pair} \in \{A : \mathrm{Set}\}\ \{B : \{x : \mathrm{El}(A)\}\ \mathrm{Set}\}\ \{a : \mathrm{El}(A)\}\ \{b : \mathrm{El}(B(a))\}\ \ \mathrm{El}(\Sigma(A,B))$

$E \in \{A : \mathrm{Set}\}\ \{B : \{x : \mathrm{El}(A)\}\ \mathrm{Set}\}\ \{C : \{z : \mathrm{El}(\Sigma(A)(B))\}\ \mathrm{Set}\}$
$\quad \{d : \{a : \mathrm{El}(A)\}\ \{b : \mathrm{El}(B(a))\}\ \ \mathrm{El}(C(\mathrm{pair}(A)(B)(a)(b)))\ \}$
$\quad \{c : \mathrm{El}(\Sigma(A)(B))\}$
$\quad\quad \mathrm{El}(C(c))$

together with the conversion rule

$$\begin{array}{l} A : \mathrm{Set}\ ,\ B : \{x : \mathrm{El}(A)\}\ \mathrm{Set}\ ,\ C : \{z : \mathrm{El}(\Sigma(A)(B))\}\ \mathrm{Set}\ , \\ d : \{a : \mathrm{El}(A)\}\ \{b : \mathrm{El}(B(a))\}\ \ \mathrm{El}(C(\mathrm{pair}(A)(B)(a)(b)))\ , \\ a : \mathrm{El}(A)\ ,\ b : \mathrm{El}(B(a)) \\ \vdash \\ E(A)(B)(C)(d)(\mathrm{pair}(A)(B)(a)(b)) = d(a)(b) \in \mathrm{El}(C(\mathrm{pair}(A)(B)(a)(b))) \end{array}$$

A fairly useful defined notion are *first* and *second projection* which in terms of E can be defined as follows

$$\begin{array}{rl} \pi_0 \equiv & [A : \mathrm{Set}]\ [B : \{x : \mathrm{El}(A)\}\ \mathrm{Set}] \\ & \quad E\ A\ B\ ([x : \mathrm{El}(A)]\ [y : \mathrm{El}(B(x))]\ A)\ ([x : \mathrm{El}(A)]\ [y : \mathrm{El}(B(x))]\ x) \end{array}$$

of type

$$\{A : \mathrm{Set}\}\ \{B : \{x : \mathrm{El}(A)\}\ \mathrm{Set}\}\ \{c : \mathrm{El}(\Sigma(A)(B))\}\ \mathrm{El}(A)$$

and

$$\begin{array}{rl} \pi_1 \equiv & [A : \mathrm{Set}]\ [B : \{x : \mathrm{El}(A)\}\ \mathrm{Set}] \\ & \quad E\ A\ B\ ([x : \mathrm{El}(A)]\ [y : \mathrm{El}(B(x))]\ B(x))\ ([x : \mathrm{El}(A)]\ [y : \mathrm{El}(B(x))]\ y) \end{array}$$

of type

$$\{A : \mathrm{Set}\}\ \{B : \{x : \mathrm{El}(A)\}\ \mathrm{Set}\}\ \{c : \mathrm{El}(\Sigma(A)(B))\}\ \mathrm{El}(B(\pi_0(A)(B)(c)))$$

and satisfy the conversion rules

$$\begin{array}{l} A : \mathrm{Set}\ ,\ B : \{x : \mathrm{El}(A)\}\ \mathrm{Set}\ ,\ a : \mathrm{El}(A)\ ,\ b : \mathrm{El}(B(a)) \\ \vdash \\ \pi_0(A)(B)(a)(b) = a \in A \end{array}$$

and

$$\begin{array}{l} A : \mathrm{Set}\ ,\ B : \{x : \mathrm{El}(A)\}\ \mathrm{Set}\ ,\ a : \mathrm{El}(A)\ ,\ b : \mathrm{El}(B(a)) \\ \vdash \\ \pi_1(A)(B)(a)(b) = b \in \mathrm{El}(B(\pi_0(A)(B)(a)(b))) \end{array}$$

respectively.

Identity Sets

Id ∈ {A : Set} {a : El(A)} {b : El(A)} Set

r ∈ {A : Set} {a : El(A)} El(Id(a)(a))

J ∈ {A : Set} {C : {x : El(A)} {y : El(A)} {z : El(Id(A)(x)(y))} Set}
{d : {x : El(A)} El(C(x)(x)(r(A)(x)))}
{a : El(A)} {b : El(A)} {c : El(Id(A)(a)(b))}
El(C(a)(b)(c))

together with the conversion rule

A : Set , C : {x : El(A)} {y : El(A)} {z : El(Id(A)(x)(y))} Set,
d : {x : El(A)} El(C(x)(x)(r(A)(x))) , a : El(A)
⊢
J(A)(C)(d)(a)(a)(r(A)(a))) = d(a) ∈ El(C(a)(a)(r(A)(a)))

According to the intuitive semantics of constructive set theory and also in all known "mathematical", i.e. models which are not term models, one can define an eliminator not only for the family [x : El(A)] [y : El(A)] Id(A)(x)(y) but also for the family [x : El(A)] Id(A)(x)(x) . This new eliminator intuitively expresses that for any a ∈ A the type Id(A)(a)(a) contains exactly one object, namely r(A)(a) .
We will call this eliminator K and it is axiomatized as

K ∈ {A : Set} {C : {x : El(A)} {z : El(Id(A)(x)(x))} Set}
{d : {x : El(A)} El(C(x)(r(A)(x)))}
{a : El(A)} {c : El(Id(A)(a)(a))} El(C(c))

together with the conversion rule

A : Set , C : {x : El(A)} {z : El(Id(A)(x)(x))} Set ,
d : {x : El(A)} El(C(r(A)(x)) ,
a : El(A)
⊢
K(A)(C)(d)(a)(r(A)(a)) = d(a) ∈ El(C(a)(r(A)(a))) .

Of course, these two eliminators are not unrelated at all and the following relation between them can be proven by straightforward application of conversion rules for J and K :

A : Set , C : {x : El(A)}{y : El(A)}{z : El(Id(A)(x)(y))} Set ,
d : {x : El(A)} El(C(x)(x)(r(A)(x))) ,
a : El(A)
⊢
J(A)(C)(d)(a)(a)(r(A)(a)) = K(A)([x : El(A)] C(x)(x))(d)(a)(r(A)(a))
∈ El(C(a)(a)(r(A)(a)))

Appendix 3

A More Readable Presentation of Constructive Set Theory

As the syntactic presentation in Appendix 2 has been a little bit bureaucratic we adopt the following notational conventions in order to improve readability.

For any set expression A we may simply write x : A instead of x : El(A) . Similarly whenever B is a set expression then we may simply write {x : A} B instead of {x : A} El(B) . Instead of t (s) we may simply write t s and assume this juxtaposition is left associativ, i.e. the expression t $s_1 \ldots s_n$ stands for t $(s_1) \ldots (s_n)$.

Then we get the following more readable formulation of constructive set theory in the logical framework .

Products of Families of Sets

$\Pi \in$ {A : Set} {B : {A} Set} Set

fun $\in$ {A : Set} {B : {A} Set} {f : {x : A} B(x)} P A B

F $\in$ {A : Set} {B : {A} Set} {C : {Π A B)} Set }
    {d : {f : {x : A} B(x)} C(fun A B f) }
    {c : Π A B}
      C c

together with the conversion rule

A : Set , B : {x : A} Set , C : {Π A B} Set ,
d : {f : {x : A} B(x)} C(λ A B f) , f : {x : A} B(x)
$\vdash$
F A B C d (fun A B f) = d f $\in$ C(fun A B f)

A fairly useful defined notion is function application which in terms of F can be defined as follows

$$\text{apply} \equiv [x : A]\,[B : \{x : A\}\,\text{Set}]\,[c : \Pi\,A\,B]\,[a : A]$$
$$F\,A\,B\,([z : \Pi\,A\,B]\,B\,a)([f : \{x : A\}\,B\,x]\,f\,a)\,c$$

for which one can derive

$$\text{apply} \in \{x : A\}\,\{B : \{x : A\}\,\text{Set}\}\,\{c : \Pi\,A\,B\}\,\{x : A\}\,B\,x\ .$$

satisfying the conversion rule

$A : \text{Set},\ B : \{x : A\}\,\text{Set}$
$f : \{x : A\}\,B(x)\,,\ a : A$
$\vdash$
$\text{apply}\,A\,B\,(\text{fun}\,A\,B\,f)\,a = f\,a \in B\,a$

Sums of Families of Sets

$$\Sigma \in \{A : \text{Set}\}\,\{B : \{x : A\}\,\text{Set}\}\,\text{Set}$$

$$\text{pair} \in \{A : \text{Set}\}\,\{B : \{x : A\}\,\text{Set}\}\,\{a : A\}\,\{b : B(a)\}\ \Sigma\,A\,B$$

$$\begin{aligned}\text{pair} \in\ & \{A : \text{Set}\}\,\{B : \{x : A\}\,\text{Set}\}\,\{C : \{c : \Sigma\,A\,B\}\,\text{Set}\}\\ & \{d : \{x : A\}\,\{y : B(x)\}\ C(\text{pair}\,A\,B\,x\,y)\}\\ & \{c : \Sigma\,A\,B\}\\ & \quad C\,c\end{aligned}$$

together with the conversion rule

$A : \text{Set}\ ,\ B : \{x : A\}\,\text{Set}\ ,\ C : \{c : \Sigma\,A\,B\}\,\text{Set}\ ,$
$d : \{x : A\}\,\{y : B(x)\}\ C(\text{pair}\,A\,B\,x\,y)\ ,$
$a : A\ ,\ b : B(a)$
$\vdash$
$E\,A\,B\,C\,d\,(\text{pair}\,A\,B\,a\,b) = d\,a\,b \in C(\text{pair}\,A\,B\,a\,b)$

First and second projection are defined as

$\pi_0 \equiv$ [A : Set] [B : {x : A} Set] E A B ([x : A] [y : B x] A) ([x : A] [y : B x] x)

of type

{A : Set} {B : {x :A} Set} {c : Σ A B} A

and

$\pi_1 \equiv$ [A : Set] [B : {x : A} Set]
E A B ([x : A] [y : B x] B(x)) ([x : A] [y : B x] y)

of type

{A : Set} {B : {x :A} Set} {c : Σ A B} B(π_0 A B c)

satisfying the conversion rules

A : Set , B : {x : A} Set , a : A , b : B $\vdash$ π_0 A B (pair A B a b) = a $\in$ A

and

A : Set , B : {x : A} Set , a : A , b : B $\vdash$ π_0 A B (pair A B a b) = b $\in$ B a

respectively.

Identity Sets

Id $\in$ {A : Set} {a : A} {b : A} Set

r $\in$ {A : Set} {a : A} Id A a a

J $\in$ {A : Set} {C : {x : A} {y : A} {z : Id A x y} Set}
{d : {x : A} C x x(r A x)}
{a : A} {b : A} {c : Id A a b}
C a b c

together with the conversion rule

$$A : \mathrm{Set}\,,\ C : \{x : A\}\ \{y : A\}\ \{z : \mathrm{Id}\ A\ x\ y\}\ \ \mathrm{Set}\,,\ d : \{x : A\}\ C\ x\ x\ (r\ A\ x)\ \ ,\ a : A$$
$$\vdash$$
$$J\ A\ C\ d\ a\ a\ (r\ A\ a)\ =\ d\ a\ \in\ C\ a\ a\ (r\ A\ a)$$

and the additional eliminator

$$\begin{aligned} J\ \in\ & \{A : \mathrm{Set}\}\ \{C : \{x : A\}\ \{\mathrm{Id}\ A\ x\ x\}\ \mathrm{Set}\} \\ & \{d : \{x : A\}\ C\ x\ (r\ A\ x)\} \\ & \{a : A\}\ \{c : \mathrm{Id}\ A\ a\ a\} \\ & \quad C\ a\ c \end{aligned}$$

together with the elimination rule

$$A : \mathrm{Set}\,,\ C : \{x : A\}\ \{\mathrm{Id}\ A\ x\ x\}\ \mathrm{Set}\,,\ d : \{x : A\}\ C\ x\ (r\ A\ x)\,,\ \ a : A$$
$$\vdash$$
$$K\ A\ C\ d\ a\ (r\ A\ a)\ =\ d\ a\ \in\ C\ (r\ A\ a)$$

Obviously J and K are related in the following way

$$A : \mathrm{Set}\,,\ \ C : \{x : A\}\{y : A\}\{\mathrm{Id}\ A\ x\ y\}\ \mathrm{Set}\,,\ \ d : \{x : A\}\ C\ x\ x\ (r\ A\ x)\,,\ \ a : A$$
$$\vdash$$
$$J\ A\ C\ d\ a\ a\ (r\ A\ a)\ =\ K\ A\ (C\ a)\ d\ a\ (r\ A\ a)\ \in\ \ C\ a\ a\ (r\ A\ a)$$

Appendix 4

Some Auxiliary Operations on Identity Sets

Substitution

sub ≡

```
[A : Set] [B : {x : A} Set] [x : A] [y : A] [z : Id A x y] [u : B x]
   apply (B x) ([u : B x] B y)
               (J  A
                  ([x : A] [y : A] [z : Id A x y] Π (B x) ([u : B x] B y))
                  ([x : A] fun (B x) ([u : B x] B x) ([u : B x] u))
                  x y z)
               u
```

of type

```
{A : Set} {B : {x : A} Set} {x : A} {y : A} {z : Id A x y} {u : B x} B y
```

with the derived conversion rule

```
A : Set , B : {x : A} Set , a : A , b : B a
⊢
sub A B a a (r A a) b  =  b  ∈  B a
```

Symmetry

sym ≡ [A : Set] J A ([x : A] [y : A] [z : Id A x y] Id A y x) ([p : Id A x x] p)

of type

```
{A : Set} {x : A} {y : A} {z : Id A x y} Id A y x
```

with the derived conversion rule

$$A : \mathrm{Set}, a : A \vdash \mathrm{sym}\ A\ a\ a\ (\mathrm{r}\ A\ a) = \mathrm{r}\ A\ a \in \mathrm{Id}\ A\ a\ a$$

Transitivity

$$\mathrm{trans} \equiv [A : \mathrm{Set}]\ [a,b,c : A]\ [p : \mathrm{Id}\ A\ a\ b]\ [q : \mathrm{Id}\ A\ b\ c]$$

$$\mathrm{sub}\ A\ ([x : A]\ \mathrm{Id}\ A\ a\ x)\ b\ c\ q\ p$$

of type

$$\{A : \mathrm{Set}\}\ \{a,b,c : A\}\ \{p : \mathrm{Id}\ A\ a\ b\}\ \{q : \mathrm{Id}\ A\ b\ c\}\ \mathrm{Id}\ A\ a\ c$$

with the derived conversion rule

$$A : \mathrm{Set},\ a : A,\ b : A,\ p : \mathrm{Id}\ A\ a\ b \vdash \mathrm{trans}\ A\ a\ b\ b\ p\ (\mathrm{r}\ A\ b) = p \in \mathrm{Id}\ A\ a\ a$$

References

[Acz] P. Aczel *The strength of Martin-Löf's intuitionistic type theory with one universe* in : The Proceedings of the Symposiums on Mathematical Logic in Oulu 1974 and Helsinki 1975, pp. 1-32 , Reports of the Department of Philosophy, Univ. Helsinki, 1977.

[dBr] N. deBruijn *Telescopic Mappings in Typed Lambda Calculus* Information and Computation 91, pp. 189-204, 1991.

[Cart] J. Cartmell *Generalized Algebraic Theories and Contextual Categories* Ph. D. Thesis, Univ. Oxford, 1978.

[Coq] Th. Coquand *Pattern Matching with Dependent Types* informal Proceedings of a workshop in Bastad, Sweden, 1992.

[Const] R. L. Constable et. al. *Implementing Mathematics with the NuPrL Development System* Prentice Hall, 1986.

[DFHHPW] G. Dowek, A. Felty, H. Herbelin, G. Huet, Ch. Paulin-Mohring, B. Werner *The coq proof assistant user's guide, version 5.6* Technical Report 134, INRIA, Rocquencourt, France, December 1992.

[Dyb] P. Dybjer *Inductive Sets and Families in Martin-Löf's Type Theory and their Set-theoretic Semantics* pp. 280-306 in Logical Frameworks, eds. G. Huet and G. Plotkin, CUP, 1991.

[FS] P. J. Freyd , A. Scedrov *Categories, Allegories* North Holland Mathematical Library 39, Noth Holland 1990.

[FMRS] P.J. Freyd, Ph. Mulry, R. Rosolini, D. Scott *Extensional PERs* Information and Computation 98, pp. 211-227, (1992).

[Hyl] J. M. E. Hyland First Steps Toward Synthetic Domain Theory Proc. of CT'90, SLNM

[HJP] M. Hyland, P. Johnstone, A. Pitts *Tripos Theory* Math. Proc. Cam. Phil. Soc. 88 , pp. 205 - 232, 1980.

[Law1] F. W. Lawvere *Equality in hyperdoctrines and comprehension schema as an adjoint* *functor* , Proc. Am. Math. Soc., 1970.

[Law2] F. W. Lawvere *Some Thoughts on the Future of Category Theory* SLNM , 1991.

[LP] Z. Luo, R. Pollack *LEGO Proof Development System : User's Manual* ECS-LFCS-92-211, Univ. Edinburgh, 1992.

[LS] J. Lambek , P. J. Scott *Introduction to Higher Order Categorical Logic* Cambridge University Press, 1985.

[Luo] Z. Luo *An Extended Calculus of Constructions* Ph. D. Thesis, ECS-LFCS-90-118, Univ. Edinburgh, 1990.

[Mag] L. Magnusson *The new implementation of ALF* in the informal proceedings of the *Logical Frameworks Workshop*, Bastad, June 1992.

[McK] J.H. McKinna *Deliverables : A Categorical Approach to Program Development in Type Theory* Ph.D. Thesis, Univ. Edinburgh , 1992 (also appeared as ECS-LFCS-92-247)

[ML1] P. Martin-Löf *An Intuitionistic Theory of Types : Predicative Part* in Logic Colloquium '73 , eds. H. Rose and J. C. Shepherdson.

[ML2] P. Martin-Löf *Constructive Mathematics and Computer Programming* in Logic, Methodology and Philosophy of Science VI , (eds. L. J. Cohen et.al.), North Holland, Amsterdam, 1982.

[ML3] P. Martin-Löf *Intuitionistic Type Theory* Bibliopolis, Naples, 1984.

[ML4] P. Martin-Löf *The Domain Interpretation of Type Theory* in the informal proceedings of *Workshop on Semantics of Programming Languages* Gotenburg, 1983.

[NPS] B. Nordström, K. Petersson and J. Smith *Programming in Martin-Löf's Type Theory. An Introduction.* Oxford University Press, 1990.

[PSH] E. Palmgren , V. Stoltenberg-Hansen *Domain Interpretations of Martin-Löf's Partial Type Theory* in APAL 48 , pp. 135-96, 1990.

[Pym] D. Pym *Proofs, Search and Computation in General Logic* Ph. D. Thesis, Univ. Edinburgh, 1990.

[Plo] G. Plotkin *Domain Theory* Lecture Notes, Univ. Edinburgh, 1983.

[Ro] H. Rogers *Theory of Recursive Functions and Effective Computability* McGrawHill 1967.

[Smi] J. Smith *On the relation between a type theoretic and a logical formulation of the theory of constructions* Ph. D. Thesis, Univ. Gothenburg, 1978.

[Str1] T. Streicher *Semantics of Type Theory. Correctness, Completeness and Independence Results.* Birkhäuser, 1991.

[Str2] T. Streicher *Dependence and independence results for (impredicative) calculi of dependent types* MSCS 1 (2), pp. 29-54, 1992.

[Str3] T. Streicher *Independence of the induction principle and the axiom of choice in the pure calculus of constructions* TCS 103 , pp. 395-408, 1992.

[vO] J. van Oosten *Exercises in Realizability* Ph. D. Thesis, Univ. Amsterdam, 1991.

Addendum

The results of 1.3 on Paulin-Mohring's Eliminator for Identity Types have been superseeded by a recent result of Martin Hofmann who could show the following

Theorem (M. Hofmann)

Paulin-Mohring's eliminator PM is definable from Martin-Löf's canonical eliminator J .

Proof : First observe that the term

```
t ≡   [A:Set]
      J A ([x,y:A][z: Id A x y]
          Id (Σ A ([u:A] Id A x u))
             (pair A ([u:A] Id A x u) x (r A x))
             (pair A ([u:A] Id A x u) y z))
          ([x:A] r (Σ A ([u:A] Id A x u)) (pair A ([u:A] Id A x u) x (r A x)))
```

is of type

```
{A:Set}{x,y:A}{z:Id A x y}
   Id (Σ A ([u:A] Id A x u))
      (pair A ([u:A] Id A x u) x (r A x))
      (pair A ([u:A] Id A x u) y z)  .
```

It is obvious from the definition of t that

```
A : Set, x : A
⊢
t A x x (r A x)  =  r (Σ A ([u:A] Id A x u)) (pair A ([u:A] Id A x u) x (r A x))
∈  Id (Σ A ([u:A] Id A x u))
      (pair A ([u:A] Id A x u) x (r A x))
      (pair A ([u:A] Id A x u) x (r A x))   .
```

Suppose A : Set , x : A , C : {y : A} (Id A x y) Set, d : C x (r A x) .
Then we may define the auxiliary family

$$D \equiv [p : \Sigma\ A\ ([u{:}A]\ \mathrm{Id}\ A\ x\ u)]\ C\ (\pi_0\ A\ ([u{:}A]\ \mathrm{Id}\ A\ x\ u))\ (\pi_1\ A\ ([u{:}A]\ \mathrm{Id}\ A\ x\ u))$$

If we further assume that d : C y z = D (pair A ([u:A] Id A x u) y z) and y : A , z : Id A x y then

```
s  ≡   sub (Σ A ([u:A] Id A x u)) D
```

```
        (pair A ([u:A] Id A x u) x (r A x))
        (pair A ([u:A] Id A x u) y z)
        (t A x y z)
        d
```

is of type

$$D\ (\text{pair } A\ ([u{:}A]\ \text{Id } A\ x\ u)\ y\ z)\ =\ C\ y\ z\ .$$

Thus we have proved that

```
A : Set , x : A , C : {y : A} (Id A x y) Set, d : C x ( r A x) , d : C y z  y : A , z : Id A x y
⊢
sub  (Σ A ([u:A] Id A x u)) D
     (pair A ([u:A] Id A x u) x (r A x))
     (pair A ([u:A] Id A x u) y z)
     (t A x y z)
     d
∈ C y z
```

Furthermore we have

```
A : Set , x : A , C : {y : A} (Id A x y) Set, d : C x ( r A x)
⊢
sub  (Σ A ([u:A] Id A x u)) D
     (pair A ([u:A] Id A x u) x (r A x))
     (pair A ([u:A] Id A x u) x (r A x))
     (t A x x (r A x))
     d
= d
∈ C y z
```

as t A x x (r A x) = r (Σ A ([u:A] Id A x u)) (pair A ([u:A] Id A x u) x (r A x)) (shown above)
and

```
sub  (Σ A ([u:A] Id A x u)) D
     (pair A ([u:A] Id A x u) x (r A x))
     (pair A ([u:A] Id A x u) x (r A x))
     r (Σ A ([u:A] Id A x u)) (pair A ([u:A] Id A x u) x (r A x))
     d
```

converts with d (cf. Appendix 4). ◊