J. Functional Programming **8** (4): 413–436, July 1998. Printed in the United Kingdom

A coherence theorem for Martin-Löf's type theory

MICHAEL HEDBERG

Programming Methodology Group, Department of Computer Science, University of Göteborg and Chalmers University of Technology, S-412 96 Göteborg, Sweden

Abstract

In type theory a proposition is represented by a type, the type of its proofs. As a consequence, the equality relation on a certain type is represented by a binary family of types. Equality on a type may be conventional or inductive. Conventional equality means that one particular equivalence relation is singled out as *the* equality, while inductive equality – which we also call *identity* – is inductively defined as the 'smallest reflexive relation'. It is sometimes convenient to know that the type representing a proposition is collapsed, in the sense that all its inhabitants are identical. Although uniqueness of identity proofs for an arbitrary type is not derivable inside type theory, there is a large class of types for which it may be proved. Our main result is a proof that any type with decidable identity has unique identity proofs. This result is convenient for proving that the class of types with decidable identities is closed under indexed sum. Our proof of the main result is completely formalized within a kernel fragment of Martin-Löf's type theory and mechanized using ALF. Proofs of auxiliary lemmas are explained in terms of the category theoretical properties of identity. These suggest two coherence theorems as the result of rephrasing the main result in a context of conventional equality, where the inductive equality has been replaced by, in the former, an initial category structure and, in the latter, a smallest reflexive relation.

Capsule Review

This paper offers new results about propositional equality in Martin-Löf's intensional type theory. The main result states that if equality on a type A is decidable, then any two proofs of equality between elements of A are equal themselves. This seemingly academic question is of immediate practical relevance for programming with dependent types.

For example, if we model many-sorted stores as `Pi l:Loc.TypeOf(l)` where `TypeOf : Loc -> Set` is a type-valued function assigning types to locations, then to establish the expected properties of updating such stores we need to know that any two proofs of location equality are equal (example due to Thomas Schreiber). The main result guarantees this under the reasonable assumption that equality of locations is decidable.

A possible obstacle for readers that are not experts in Martin-Löf type theory is the fact that (following recent tradition) the paper uses intentional type theory in which propositional equality is witnessed by proof objects and in which (unlike in extensional type theory) type checking is decidable, but that in the context of extensional type theory (which is perhaps better known) the results in the paper are trivial.

1 Introduction

1.1 Type theory

Martin-Löf's type theory (Martin-Löf, 1972; Nordström *et al.*, 1990) may be seen as a framework for expressing constructive mathematical reasoning. The interactive proof assistant ALF (Augustsson *et al.*, 1992; Altenkirch *et al.*, 1994; Magnusson and Nordström, 1994) can be used to implement a formalization of type theory and to develop formal type theory proofs. The fundamental ingredients of type theory are those of *set*, *element of a set*, *family of sets* and *dependently typed*[1] *function.*

By the propositions-as-types principle, a proposition is represented by a set, the set of its (constructive) proofs, and the logical constants, i.e. the quantifiers and the connectives, are represented by corresponding set forming operations in accordance with the Curry–Howard analogy or the Brouwer–Heyting–Kolmogorov interpretation. When we later use familiar terms such as *set*, *relation* or *category*, they should be understood in their type theoretic sense, which in most cases should be evident by the propositions-as-types principle.

Some of our concepts will be defined by quantifying over the totality of all sets or the totality of all relations on a fixed set. This should not be mistaken for impredicative quantification, but should be understood in the light of the distinction between *set* and *type*. In fact, the basic machinery of type theory, as it will be presented here, may be summarized in the following three principles: the collection of types is closed under indexed product; the collection of sets form a type; and every set is a type. We may, using a naive symbolism, depict the relation between set and type as follows: $set \in type$ and $set \subseteq type$.

Among different versions of Martin-Löf's type theory available in the literature, the present version corresponds best to the theory decribed in part III of Nordström *et al.* (1990). The two most striking features of this version of type theory as opposed to other versions are, first, the ample amount of type information explicitly given in the terms and, secondly, the absence of the equality reflection rule that would allow one to infer the *judgemental* equaltity between two terms from the corresponding *propositional* equality. *In other versions of type theory the problems we are addressing may disappear completely.*

1.2 Data irrelevance and collapsed sets

When a construction is formalized, it must sometimes be supplied with data that makes the construction legal, without affecting the final result. We point at two different reasons why a function may be constant in one of its arguments. On the one hand, the constancy may depend upon the behaviour of the function or on the notion of equality used to compare output data. On the other hand, the constancy may have nothing to do with the behaviour of the function or the equality used on

[1] The type of the result of a function application may depend upon the values given as arguments to the function, and in an m-place function $f(x_1,\ldots,x_m)$ or family of sets $A(x_1,\ldots,x_m)$, the range of the variable x_i may depend upon the values assigned to the preceding variables $x_1, \ldots, x_{i-1}$.

the output data, but with the *type of the irrelevant data*. For instance, any function defined on a singleton set must be constant regardless of how we 'measure' equality in the target set. Obviously, the knowledge that sets on a certain form are collapsed, in the sense of having at most one element, may simplify irrelevance proofs that without this knowledge would have to rely on an analysis of the context in which the irrelevant data appear.

1.3 Inductive equality and conventional equality

The statement of an irrelevance theorem must use some notion of equality to compare (syntactically) different outputs, and the statement that a set is collapsed must use some notion of equality with respect to which all elements of the set are stated to equal each other. We point out two methods for dealing with equality.

First, *inductive equality*, where equality on a set is introduced by means of an inductive definition with reflexivity as introduction rule. Unless otherwise explicitly indicated, we use the term *identity* to mean equality inductively defined by reflexivity. Secondly, *conventional equality*, which does not assume the possibility of extracting a notion of equality from an arbitrary set, and thus demands equality to be defined wherever it is used. Bishop (1967) emphasizes the conventionality of equality, which seems to be the only way in which general quotient sets can be *constructed* rather than *postulated*. Conventional equality may be systematized by developing a theory of *setoids*, where a setoid is a set together with an equivalence relation.

Remark. The relationship between the notions of *set* and *setoid* in type theory is, at least from a formal point of view, analogous to that between the notions of *preset* and *set* used by Greenleaf (1980) in an analysis of Bishop's notion of *set*. In the light of this analogy, Greenleaf's view

> The distinction between sets, which carry an equality predicate, and presets (1.2), which do not, is a central one in LCST. The novelty of our analysis of the paradoxes comes from the recognition that an arbitrary preset carries no underlying identity predicate (3). The paradoxes of Russell (3.4) and Cantor (4.1) arises from an unwarranted belief in the existence of an identity predicate. (Greenleaf, 1980, p. 215)

on presets and equality seems to be an argument against inductive equality on arbitrary sets. We prefer, however, to see this as a breakdown of the analogy between 'set in type theory' and 'preset'. There is no logical problem with inductive equality, but rather a pragmatic one – one has to remember that inductive equality on a function type does not coincide with 'ordinary' pointwise equality.

1.4 Categories in connection with type theory

Recall the informal definition of a category as a system consisting of

- a class of *objects*,
- and, for each pair of objects A and B, a set of *arrows from A to B*,
- and, for each object A, a designated arrow $1_A = 1$ from A to A, the *unit arrow*,

- and, for each triple A, B and C of objects, a binary operation *composing* an arbitrary arrow f from B to C with an arbitrary arrow g from A to B to produce an arrow $f \cdot g$ from A to C,
- such that composition of arrows is associative with the unit arrows acting as left and right unit, i.e. such that $(f \cdot g) \cdot h = f \cdot (g \cdot h)$ and $1 \cdot f = f \cdot 1 = f$ for arbitrary arrows f, g and h of the appropriate sorts.

The Generalized Algebraic Theories (GAT) of Cartmell (1986) may be used to present the axiom system for an abstract category. In doing so, there is a choice concerning the treatment of arrow equality. One alternative is to express the arrow equations, such as associativity of arrow composition, in terms of the built in equality that goes with every sort in the language of GAT, but there is also the option of introducing a third sort for (proofs of) arrow equality. The first alternative would, in the context of type theory, correspond to expressing arrow equations in terms of definitional equality, and would lead to a too narrow concept. Therefore, we follow the path indicated by the second alternative. As to the nature of the sorts used for objects, arrows and arrow equality, we will be a little vague concerning the objects, but usually insist that the arrows (with fixed source and target) form a set, and that the proofs that two given arrows are equal also form a set.

1.5 Categories and preordered sets

It goes almost without saying that the (constructive) notion of a preordered set coincides with that part of a small category which does not involve arrow equality (here 'small' means that the objects form a set). Thus, to define a preorder on a set is the same as defining the arrow sets, the unit arrows and the arrow composition[2] operators of a category yet to be completed. In terms of the informal definition of category in section 1.4, this corresponds to starting the category construction with a *set* of objects and then breaking off the construction when we reach the 'such that' part.

Filling the gap between a preordered set and a small category, the objects of which are precisely the elements of the set, consists of defining equivalence relations on the arrow sets and proving the category axioms, which in addition to associativity and the left and right unit laws also include the compatibility between arrow composition and arrow equality. The gap can always be filled by using the always true relation as arrow equality, and we refer to the resulting category as the category with *collapsing arrow equality*.

Thus a small category is nothing but a preordered set with 'nice' equalities on the preorder proof sets and a preordered set is nothing but a small category where the arrow equality has been stripped off.

[2] Transitivity and composition take the same arguments, but in a different order. The difference is analogous to that between sequential $g;f$ and functional $f \circ g$ composition of functions, both defined to equal $(\lambda x)f(g(x))$.

1.6 The category of category structures on a set

The balance between a set and a category whose objects are precisely the elements of the set is referred to as a *category structure* on the set. A *category structure homomorphism* is a functor defined to have identity object action. We can turn the category structures on a set into a category using category structure homomorphisms as arrows. Since these are defined to have identity object action, we can, without using natural transformations, define two category structure homomorphisms (of the same type) to be equal if they always send the same arrow to equal arrows.

1.7 Initial category structures

An *initial category structure* on a set is the same as an initial object in the category of category structures on the set. A trivial consequence of the definition is that if a set has an initial category structure, then for any category, any family of objects in the category labelled by elements of the set can in a unique way be extended to a functor defined on the initial category.

1.8 The identity category

Central to our work is the view, put forward by Hofmann and Streicher (1994), of the identity proofs (that two elements of a certain set are identical in the sense of inductive equality) as the arrows in a category, the *identity category* associated with the set, where inductive equality is used again as arrow equality.

1.9 Identity in type theory without inductive equality

Without inductive equality available at any set, it is difficult to *define* a nontrivial equality on an arbitrary set. We can, however, *specify* identity as a relation capable of mimicking certain aspects of inductive equality. To this end, we will introduce two classes of relations. A relation on a set is a *logical identity* if it is a smallest reflexive relation, that is, it is reflexive and contained[3] in any other reflexive relation on the set. A relation on a set is a *categorical identity* if it is the preorder underlying an initial category structure on the set (cf section 1.5).

Observation

- *Any categorical identity is also a logical identity.*[4] Since any preorder can be extended to a category structure, it follows from the definition that any categorical identity is also a *smallest preorder*. Thus (by an easy exercise), it is a smallest reflexive relation.

[3] If R_1 and R_2 are relations on a certain set, we say that R_1 is *contained* in R_2 if any two elements related by R_1 are also related by R_2.

[4] Note that proving the converse in the natural way would depend upon having access to a smallest arrow equality compatible with a given preorder.

- *If we have inductive equality on a set then this is also a logical identity*. This will be evident from the elimination rule for inductive equality to be stated in section 2.
- *If we have inductive equality on a set and, moreover, inductive equalities on the corresponding identity proof sets, then the inductive equality on the original set is also a categorical identity*. After constructing the identity category, it remains only to verify its initiality (in the category of category structures).

1.10 Definitions

A set is *decidable* if it contains an element, or if the assumption of an element would yield an element of the empty set. We may also refer to a decidable set as a D-set.

A set is *collapsed* if all its elements are identical. We may also refer to a collapsed set as a C-set.

A set has *decidable identities* if, for any two of its elements, the set of proofs that they are identical is decidable. We may also refer to a set with decidable identities as a DI-set.

A set has *collapsed identity sets* if, for any two of its elements, the set of proofs that they are identical is collapsed. We may also refer to a set with collapsed identity sets as a set with unique identity proofs, or as a CI-set.

1.11 Unicity of identity proofs

With a possible exception for the indexed product, the elimination rule for a set former in type theory may be constructed from the formation and introduction rules according to a uniform pattern which is made explicit in Dybjer (1991, 1997). Thus we may talk about the *standard elimination rule* associated with a set former. Although one might expect the identity sets to be collapsed, it is impossible to prove in general that they are, as long as type theory is restricted to the standard elimination rules. This negative result was proved by Hofmann and Streicher (1994) using a groupoid interpretation of type theory.

Thus, uniqueness of identity proofs is not a general principle of type theory, and can be established only in special cases unless explicitly added as an axiom. Hofmann's (Hoffman and Streicher, 1994) main results concern what you *cannot do without* a unicity-of-identity-proofs axiom and what you *can do with* it, but he also contributes to our concern of what you *can do without* it (Hofmann, 1995a). Hofmann (1995b) shows that the empty set, the singleton set and the set of natural numbers all are CI-sets. He also shows that the class of CI-sets is closed under the formation of identity set and indexed sum.

1.12 Monoidal coherence and the discrete category

In the work on monoidal coherence by Beylin and Dybjer (1996), the notion of 'the set of natural numbers seen as a category' plays a central part. On the informal level this category may be thought of as the *discrete category* defined to have exactly one

or no arrow from m to n, depending on whether or not m and n are the same natural number. In their formalization, however, the arrows are defined to be the identity proofs.

This means that what on the informal level is justified by the discrete arrow sets being collapsed is, in the formalization, reflected if not by the explicit use of a uniqueness-of-identity-proofs lemma, then by proofs on the fly of the relevant instances of such a lemma.

One interpretation of Hofmann's negative result (Hofmann and Streicher, 1994) is that we cannot in general let the identity category play the part of the discrete category, since we cannot prove (in type theory) that the arrow sets are collapsed. In this context, the DI⊆CI-theorem stated in the next section shows that whenever the discrete category on a set makes sense, i.e. when the set has decidable identities, the identity category is equivalent to the discrete category.

1.13 The present result

The present result is a generalization of Hofmann's proof, which is based on a decision procedure for equality, that the set of natural numbers has unique identity proofs.

- **DI⊆CI-theorem.** *Any set with decidable identities has collapsed identity sets.* We give a formal proof in the fragment of type theory generated by the indexed product, the binary sum, the empty set and the identity set (and the corresponding introduction and standard elimination rules).

We note in passing that the DI⊆CI-theorem combined with the negative result (Hofmann and Streicher, 1994) on uniqueness of identity proofs shows that there can be no decision procedure expressed in type theory for the identity on an arbitrary set. As an application of the DI⊆CI-theorem we will give an informal proof that the class of DI-sets is closed under indexed sum.

- **ΣDI-theorem.** *The sum of a family of DI-sets indexed by elements of a DI-set is again a DI-set.* This is proved informally in the fragment of type theory generated by the indexed product, the indexed sum, the binary sum, the empty set and the identity set (and the corresponding introduction and standard elimination rules).

To isolate certain aspects of the proof of the DI⊆CI-theorem, namely *decidability* and the *category theoretical view on identity*, we give two theorems in type theory without inductive equality that in different ways can be combined with portions of the theory of inductive equality to reproduce the DI⊆CI-theorem. The first says that a decidable categorical identity has, in a certain sense, collapsed identity proof sets. The second implies that, in the presence of decidability, the distinction between logical and categorical identity disappears.

- **Coherence theorem 1.** *If a set has an initial category structure with decidable arrow sets, then these are collapsed with respect to arrow equality.* We give an

informal proof in type theory without identity sets, but with indexed products, indexed sums, binary sums and the empty set.

- **Coherence theorem 2.** *If a set has a decidable smallest reflexive relation then this relation is a preorder for which the corresponding category with collapsing arrow equality is an initial category structure.* We give an informal proof in type theory without identity sets, but with indexed products, indexed sums, binary sums and the empty set.

1.14 Paper Organization

The paper is organized as follows. In section 2 we introduce ALF notation and informal type theory notation for the sets that we shall use. Section 3 develops some of the theory of inductive equality by giving a detailed proof of the DI⊆CI-theorem in parallel with the corresponding ALF code, as well as an informal proof of the ΣDI-theorem. Finally, in section 4 we give proof sketches for the two coherence theorems, and show how they may be employed in alternative proofs of the DI⊆CI-theorem.

2 Notational issues for types and certain sets

When we present the rules associated with the basic set forming operations, functions taking elements of sets as input and giving elements of sets as outputs are not enough. We also need functions that give sets as outputs. One reason for us to introduce a level of types above the level of sets is that ALF is organized in this way. Another is to make sense of the big quantifications mentioned in section 1.1 and used, for instance, in the definition of logical identity in section 1.9.

2.1 Types and ALF

The formal kernel of ALF may be described as a dependently typed $\lambda\beta\eta$-calculus with a distinguished type, the type of sets, and a distinguished family of types over the type of sets that to each set associates the type of its elements. We give only informal explanations of the notations associated with types, the purpose being to render some readability to the ALF code that we shall present.

Product type. If α is a type and β is an expression possibly containing occurrences of the variable x such that $(x = a)\beta$, i.e. the result of substituting a for the free occurrencies of x in β with the usual renaming of bound variables to avoid clashes, is a type for an arbitrary object a of type α, then $(x : \alpha)\beta$ is the type of functions that to an arbitrary object a of type α assigns an object of type $(x = a)\beta$.

Application. If α, β and x are as in the previous paragraph and b is a function of type $(x : \alpha)\beta$ and a is an object of type α, then $b(a)$ is the object of type $(x = a)\beta$ obtained by applying b to the argument a.

Abstraction. If α, β and x are as above, and b is an expression possibly containing occurrencies of the variable x such that $(x = a)b$ is an object of type $(x = a)\beta$ for

an arbitrary object a of type α, then $[x]b$ is the function of type $(x : \alpha)\beta$ that to an arbitrary object a of type α assigns the object $(x = a)b$ of type $(x = a)\beta$.

Iterated product, abstraction and application.

The notation	is used in favour of
$(x_1 : \alpha_1; \ldots; x_n : \alpha_n)\beta$	$(x_1 : \alpha_1) \ldots (x_n : \alpha_n)\beta$
$[x_1, \ldots, x_n]b$	$[x_1] \ldots [x_n]b$
$b(a_1, \ldots, a_n)$	$b(a_1) \ldots (a_n)$

In the iterated product, the variable x_i may be omitted if it does not appear in any subsequent α_k, $k = i + 1, \ldots, n$ or in β. In the iterated product, again, adjacent identical types need not be duplicated, so, for instance, $(\ldots\ x : \alpha; y : \alpha\ \ldots)\beta$ may be abbreviated $(\ldots\ x, y : \alpha\ \ldots)\beta$.

The type of sets and the type of elements. The symbol Set is used for the type of sets and, when A is a set, El(A) is the type of its elements. In communications with ALF, the El symbol is suppressed altogether and the colon is pretty printed as $\in$.

We may sometimes emphasize the distinction between set and type by writing $\in$ for 'is element of the set' and : for 'is object of the type'.

We may prefer the notation	in favour of
$a \in A$	$a : \mathsf{El}(A)$
$\alpha \to \beta$	$(\alpha)\beta$
$f(x) : \beta\ [x : \alpha]$	$f : (x : \alpha)\beta$

2.2 Particular sets

The indexed product, $\Pi(A, B)$. If A is a set and $B(x)[x \in A]$ is a family of sets, then $\Pi(A, B)$ is the set of functions that to an arbitrary element a of A assigns an element of $B(a)$. More precisely, the elements of $\Pi(A, B)$ are codes representing functions of the type $(x \in A)B(x)$. If the encoding of a function $b(x) \in B(x)$ $[x \in A]$ is denoted $\lambda_{AB}(b)$ and the decoding operation, which is a function of the type $(f \in \Pi(A, B);\ a \in A)B(a)$, is denoted app_{AB}, then the appropriate relationship between the decoding and encoding operations is expressed by the β conversion rule $app_{AB}(\lambda_{AB}(b), a) = b(a)$.

Prod $\in$ $(A \in$ **Set**; $B \in (A)$ **Set**$)$ **Set**
  lam $\in$ $(A \in$ **Set**; $B \in (A)$ **Set**; $b \in (x \in A)\, B(x))$ **Prod**(A, B)
app $\in$ $(A \in$ **Set**; $B \in (A)$ **Set**; $fn \in$ **Prod**(A, B); $a \in A)\ B(a)$
  app$(A, B,$ **lam**$(\_, \_, b), a)$ $\equiv$ $b(a)$

Remark. We will in general prefer the product *type* to the product *set* for representing universal quantification. In the presence of product types, the main use that we shall make of the product set is to translate a family of *types* into a family of *sets* to which an elimination rule may be applied, as in section 3.2.1.

The identity set, $I_A(a, b)$. If A is a set and a and b are two of its elements, then $I_A(a, b)$ is the set of proofs that a and b are identical. The proof of identity by reflexivity is

denoted $r_A(x) \in I_A(x,x)\ [x \in A]$ and, if $C(x,y,z)[x,y \in A;\ z \in I_A(x,y)]$ is a family of sets, then, by I-elimination, a function $c(x,y,z) \in C(x,y,z)[x,y \in A; z \in I_A(x,y)]$ may be defined by stipulating the values $c(x,x,r_A(x)) \in C(x,x,r_A(x))[x \in A]$.

$$\begin{array}{l} \mathbf{I} \in (A \in \mathbf{Set};\\ \quad a,b \in A\\ \quad)\,\mathbf{Set}\\ \mathbf{ref} \in (A \in \mathbf{Set};\\ \quad a \in A\\ \quad)\,\mathbf{I}(A,a,a) \end{array} \qquad \begin{array}{l} \mathbf{elimI} \in (A \in \mathbf{Set};\\ \quad a,a' \in A;\\ \quad C \in (x,y \in A;\, \mathbf{I}(A,x,y))\,\mathbf{Set};\\ \quad c \in (x \in A)\,C(x,x,\mathbf{ref}(A,x));\\ \quad p \in \mathbf{I}(A,a,a')\\ \quad)\,C(a,a',p)\\ \mathbf{elimI}(A,\_,a',C,c,\mathbf{ref}(\_,\_)) \equiv c(a') \end{array}$$

Remark. We use the abbreviation $a \simeq b$ for the assertion that a and b are identical elements, i.e. that the set $I_A(a,b)$ is inhabited.[5] Thus, a proof of $a \simeq b$ is literally the same as an element of $I_A(a,b)$.

The indexed sum, $\Sigma(A,B)$. If A is a set and $B(x)[x \in A]$ is a family of sets, then $\Sigma(A,B)$ is the set of ordered pairs with one component a in A and the other in $B(a)$. More precisely, the elements of $\Sigma(A,B)$ are codes representing such pairs. We write (a,b) for the element of $\Sigma(A,B)$ that corresponds to elements a of A and b of $B(a)$ and, if $C(z)[z \in \Sigma(A,B)]$ is a family of sets, then, by Σ-elimination, a function $c(z) \in C(z)[x \in \Sigma(A,B)]$ may be defined by stipulating the values $c((x,y)) \in C((x,y))[x \in A; y \in B(x)]$.

$$\begin{array}{l} \mathbf{Sum} \in (A \in \mathbf{Set};\ B \in (A)\,\mathbf{Set})\,\mathbf{Set}\\ \mathbf{pair} \in (A \in \mathbf{Set};\\ \quad B \in (A)\,\mathbf{Set};\\ \quad a \in A;\\ \quad b \in B(a)\\ \quad)\,\mathbf{Sum}(A,B) \end{array} \qquad \begin{array}{l} \mathbf{split} \in (A \in \mathbf{Set};\\ \quad B \in (A)\,\mathbf{Set};\\ \quad C \in (\mathbf{Sum}(A,B))\,\mathbf{Set};\\ \quad c \in (x \in A;\, y \in B(x))\,C(\mathbf{pair}(A,B,x,y));\\ \quad p \in \mathbf{Sum}(A,B)\\ \quad)\,C(p)\\ \mathbf{split}(A,B,C,c,\mathbf{pair}(\_,\_,a,b)) \equiv c(a,b) \end{array}$$

The binary sum, $A+B$. If A and B are sets, then $A+B$ is their disjoint union. An element of $A+B$ is an element of A or an element of B together with information about which of the two alternatives was used. The canonical injections are denoted $i_{AB} : A \to A+B$ and $j_{AB} : B \to A+B$, respectively, and if $C(z)[z \in A+B]$ is a family of sets, then by $+$-elimination, a function $c(z) \in C(z)[z \in A+B]$ may be defined by stipulating the values $c(i(x)) \in C(i(x))[x \in A]$ and $c(j(y)) \in C(j(y))[y \in B]$.

$$\begin{array}{l} \mathbf{Plus} \in (A,B \in \mathbf{Set})\,\mathbf{Set}\\ \mathbf{i} \in (A,B \in \mathbf{Set};\\ \quad a \in A\\ \quad)\,\mathbf{Plus}(A,B)\\ \mathbf{j} \in (A,B \in \mathbf{Set};\\ \quad b \in B\\ \quad)\,\mathbf{Plus}(A,B) \end{array} \qquad \begin{array}{l} \mathbf{when} \in (A,B \in \mathbf{Set};\\ \quad C \in (\mathbf{Plus}(A,B))\,\mathbf{Set};\\ \quad f \in (a \in A)\,C(\mathbf{i}(A,B,a));\\ \quad g \in (b \in B)\,C(\mathbf{j}(A,B,b));\\ \quad p \in \mathbf{Plus}(A,B)\\ \quad)\,C(p)\\ \mathbf{when}(A,B,C,f,g,\mathbf{i}(\_,\_,a)) \equiv f(a)\\ \mathbf{when}(A,B,C,f,g,\mathbf{j}(\_,\_,b)) \equiv g(b) \end{array}$$

[5] For linguistical reasons we may pass from a set to a proposition using the expression 'A is inhabited', which is defined to denote the proposition which is proved by exhibiting an element of the set A.

The empty set, N_0**.** This set has no elements and, if $C(z)[z \in N_0]$ is a family of sets, we may, by N_0-elimination, introduce a function $c(z) \in C(z)[z \in N_0]$.

$$\mathbf{N}_0 \in \mathbf{Set} \qquad \mathbf{case}_0 \in (C \in (\mathbf{N}_0)\mathbf{Set};\ p \in \mathbf{N}_0)\ C(p)$$

The decidability set, $\mathscr{D}(A) = A + (\Pi x \in A)N_0$**.** This is not a new set, but the formal expression of the definition of 'decidable set' given in section 1.10.

$$\begin{aligned} \mathrm{Dec} &\in (A \in \mathbf{Set})\ \mathbf{Set} \\ \mathrm{Dec} &\equiv [A]\mathbf{Plus}(A, \mathbf{Prod}(A, [h]\mathbf{N}_0)) \end{aligned}$$

Remark. When a family over the set A is given by abstraction we may write $(\Pi x \in A)B$ and $(\Sigma x \in A)B$ alternatively to $\Pi(A, [x]B)$ and $\Sigma(A, [x]B)$, respectively.

3 DI⊆CI in the context of inductive equality

In this section we give a completely formalized and mechanized proof of the DI⊆CI-theorem in the context of inductive equality. We also show how the DI⊆CI-theorem may be employed in a proof of the ΣDI-theorem (that the class of DI-sets is closed under indexed sum).

3.1 Intuition

We occasionally adhere to some of the notational conventions of category theory. For instance, '$f : A \overset{\mathscr{C}}{\longrightarrow} B$' means the same as '$f$ is an arrow (morphism) from (the object) A to (the object) B in (the category) $\mathscr{C}$'. The category label $\mathscr{C}$ may be omitted. If an arrow drawn in a diagram has a label attached to it, then the label is usually a name for the arrow and not a name for a category.

In connection with functors, we may in notation confuse the object action and the arrow action and say that a functor F sends an arrow $f : A \longrightarrow B$ to an arrow denoted '$F(f) : F(A) \longrightarrow F(B)$', rather than the more pedantic '$F_1(A, B, f) : F_0(A) \longrightarrow F_0(B)$', where F_0 and F_1 are the object and arrow action of F, respectively.

3.1.1 The identity category

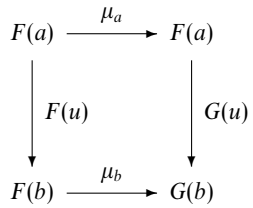
Central to our proof is the view of the equality proofs as arrows of a category. This means that for an arbitrary set A, the elements of A may be seen as the objects of a category, *the identity category on A*, where an arrow from the object (i.e. element of A) a to the object (i.e. element of A) b is the same as a proof that $a \simeq b$, i.e. an element of $I_A(a, b)$, and where two arrows (of the same type) are considered as equal if they are identical (in the sense of inductive equality). One can show that indeed this gives us, not only a category, but a category where every arrow has an inverse with respect to arrow composition. We prove the relevant parts of this fact when needed, cf. section 3.2.1.

The identity category associated with a set is in a certain sense the 'smallest' among all categories where the objects are precisely the elements of the set, i.e. it

represents an initial category structure. Some of the things we do with identity can be explained in terms of the universal property of the identity category, some of which we list below.

- The operations identity coercion and identity mapping, which will be introduced later.
- The fact that arrows in the identity category are invertible.
- The following naturality lemma, a special case of which will be proved and used in the proof of the DI⊆CI-theorem.

  If $\mu_x : F(x) \xrightarrow{\mathscr{C}} G(x)$ $[x \in A]$ is a family of arrows (where F and G are functors from the identity category on A to some category $\mathscr{C}$) then μ is a natural transformation, meaning that the following diagram commutes in $\mathscr{C}$:

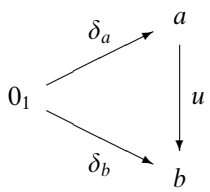


Naturality diagram ($u \in I_A(a, b)$).

3.1.2 *The singleton set has unique identity proofs*

To illustrate the use of naturality, we treat the case of the singleton set N_1 which has one element, $0_1 \in N_1$. The elimination rule for the singleton set states, for an arbitrary family of sets $C(x)$ $[x \in N_1]$, that a function $c(x) \in C(x)$ $[x \in N_1]$ may be defined by stipulating the value $c(0_1) \in C(0_1)$. This rule, together with identity being reflexive, immediately gives us a family of arrows $\delta_x : 0_1 \longrightarrow x$ $[x \in N_1]$ in the identity category associated with N_1. (Recall that '$f : a \longrightarrow b$' in this context is a synonym for '$f \in I_{N_1}(a, b)$'.)

The family of arrows δ would be a natural transformation from a constant functor to the identity functor, if only the diagram



Commutative(?) diagram of identity proofs.

were commutative for arbitrary elements $a, b \in N_1$ and $u \in I(a, b)$. Assuming for the moment the naturality of δ, this means that $u \cdot \delta_a \simeq \delta_b$ so that, by the groupoid laws, if we have two proofs u and v such that $a \simeq b$, they must both be identical to $\delta_b \cdot (\delta_a)^{-1}$. The naturality of δ follows immediately by identity elimination. Note how

naturality here is used almost verbatim in the same way as in the work of Beylin and Dybjer mentioned in section 1.12.

3.1.3 The general case

In the case of an arbitrary set A with decidable identities, decidability is used to define a family of *constant* functions $v_{xy} : I_A(x, y) \to I_A(x, y)$ $[x, y \in A]$ to which the naturality lemma can be seen to apply – for a fixed element a of A, we may show the family of functions $v_{ax} : I_A(a, x) \to I_A(a, x)$ $[x \in A]$ to be a natural transformation from the Yoneda functor to itself. The naturality equation gives, combined with the groupoid laws, a family of *left inverses* to the functions v_{xy}. This leads to a situation where we have a set $I_A(x, y)$ on which there is defined a constant function with a left inverse. Clearly, such a set must be collapsed.

3.2 Proof of the DI⊆CI-theorem

Here we give a formal proof of the DI⊆CI-theorem, organized as follows. First, some elementary identity theory, secondly, the definition of a constant 'endo function' on a decidable set, (constancy lemma), thirdly, a proof that a set on which there is defined a constant function with a left inverse must be collapsed, (collapse lemma), fourthly, the definition of a family of left inverses to a family of functions $v_{xy} : I(x, y) \to I(x, y)[x, y \in A]$, (left inverse lemma), and finally, the combination of the three lemmas in a proof of the DI⊆CI-theorem.

3.2.1 Identity theory

Before we define the groupoid operations on the identity sets and prove the groupoid law that we shall need, we introduce the operations *identity coercion* and *identity mapping* corresponding to the logical rules

$$\frac{x \simeq y \qquad P(x)}{P(y)} \qquad\qquad \frac{x \simeq y}{f(x) \simeq f(y)}$$

which express that predicates are stable under identity and that any function (with constant codomain) preserves identity, respectively.

Identity coercion and identity mapping

Identity coercion. If A is a set and if B is a family of sets over A, we can, using I-elimination, define a family of functions

$$B^*_{xy} : I(x, y) \to B(x) \to B(y) \quad [x, y \in A]$$

subject to the definitional equality

$$B^*_{xx}(r_A(x), b) = b \quad [x \in A;\ b \in B(x)].$$

(A *definitional equality* is an equation that follows from the defining equations appearing as abbreviations or as instances of the elimination rules.) This is a place

where we, in the formal proof, need the product *set* to apply the elimination rule for identity.

$$\begin{array}{l}\text{coercI} \in (A \in \mathbf{Set};\ a, a' \in A;\ B \in (A)\mathbf{Set};\ u \in \mathbf{I}(A, a, a');\ b \in B(a))\ B(a') \\ \quad \text{coercI} \equiv \\ \qquad [A, a, a', B, u, b] \\ \qquad\quad \mathbf{app}(B(a), \\ \qquad\qquad [h]B(a'), \\ \qquad\qquad \mathbf{elimI}(A, a, a', [x, y, z]\mathbf{Prod}(B(x), [h]B(y)), [x]\mathbf{lam}(B(x), [h]B(x), [y]y), u), \\ \qquad\qquad b)\end{array}$$

Identity mapping. If A and B are sets and f is a function from A to B, we can, using I-elimination, define a family of functions

$$\widehat{f}_{xy} : I_A(x, y) \to I_B(f(x), f(y)) \quad [x, y \in A],$$

subject to the definitional equality

$$\widehat{f}_{xx}(r_A(x)) = r_B(f(x)) \quad [x \in A].$$

$$\begin{array}{l}\text{mapI} \in (A, B \in \mathbf{Set};\ a, a' \in A;\ f \in (A)B;\ u \in \mathbf{I}(A, a, a'))\ \mathbf{I}(B, f(a), f(a')) \\ \quad \text{mapI} \equiv [A, B, a, a', f, u]\mathbf{elimI}(A, a, a', [x, y, z]\mathbf{I}(B, f(x), f(y)), [x]\mathbf{ref}(B, f(x)), u)\end{array}$$

Category theoretical interpretation. In the language of category theory, identity coercion $B^*_{xy} : I(x, y) \to B(x) \to B(y)$ $[x, y \in A]$ can be shown to constitute the arrow action of a functor from the identity category on A to the category of sets and functions[6], where the object action is given by the family $B(x)[x \in A]$ of sets over A, and identity mapping $\widehat{f}_{xy} : I_A(x, y) \to I_B(f(x), f(y))$ $[x, y \in A]$ can be shown to constitute the arrow action of a functor from the identity category on A to the identity category on B where the object action is given by the function f from A to B.

Groupoid structure on the identity sets

Groupoid operations. For an arbitrary set A, we define the groupoid operations on identity proofs. The unit arrow, written $1_a \in I(a, a)[a \in A]$, is the proof by reflexivity, $r_A(a)$. Arrow composition, written $u \cdot v \in I(a, c)[a, b, c \in A; u \in I(b, c); v \in I(a, b)]$, is defined using identity coercion with the family of sets $I(a, z)$ $[z \in A]$. The inverse arrow, written $u^{-1} \in I(b, a)[a, b \in A; u \in I(a, b)]$, is defined using I-elimination with the family of sets $I(y, x)$ $[x, y \in A;\ z \in I(x, y)]$. These operations are subject to the definitional equalities $1_b \cdot u = u[a, b \in A;\ u \in I(a, b)]$ and ${1_a}^{-1} = 1_a[a \in A]$.

$$\begin{array}{l}\text{cmpI} \in (A \in \mathbf{Set};\ a, b, c \in A;\ u \in \mathbf{I}(A, b, c);\ v \in \mathbf{I}(A, a, b))\ \mathbf{I}(A, a, c) \\ \quad \text{cmpI} \equiv [A, a, b, c, u, v]\text{coercI}(A, b, c, [x]\mathbf{I}(A, a, x), u, v) \\ \text{invI} \in (A \in \mathbf{Set};\ a, b \in A;\ u \in \mathbf{I}(A, a, b))\ \mathbf{I}(A, b, a) \\ \quad \text{invI} \equiv [A, a, b, u]\mathbf{elimI}(A, a, b, [x, y, z]\mathbf{I}(A, y, x), [x]\mathbf{ref}(A, x), u)\end{array}$$

[6] Two functions are considered as equal if they always send the same input to identical outputs.

A groupoid law. To define a family of functions

$$invrI_{Axy}(z) \in (z \cdot z^{-1} \simeq 1) \quad [x, y \in A; z \in I(x, y)]$$

proving that the inverse arrow is a right inverse with respect to arrow composition, we use I-elimination with the family of sets $I(z \cdot z^{-1}, z)$ $[x, y \in A;\ z \in I(x, y)]$, which leaves us, after simplification, with inhabiting the set $I(r(x), r(x))$ for an arbitrary element x of A. Such an inhabitant is of course given by the reflexivity of identity.

$$\begin{array}{l}
\text{invrI} \in (A \in \mathbf{Set};\\
\qquad a, b \in A;\\
\qquad u \in \mathbf{I}(A, a, b)\\
\qquad)\,\mathbf{I}(\mathbf{I}(A, b, b), \text{cmpI}(A, b, a, b, u, \text{invI}(A, a, b, u)), \mathbf{ref}(A, b))\\
\quad \text{invrI} \equiv\\
\qquad [A, a, b, u]\\
\qquad\quad \mathbf{elimI}(A,\\
\qquad\qquad a,\\
\qquad\qquad b,\\
\qquad\qquad [x, y, w]\mathbf{I}(\mathbf{I}(A, y, y), \text{cmpI}(A, y, x, y, w, \text{invI}(A, x, y, w)), \mathbf{ref}(A, y)),\\
\qquad\qquad [x]\mathbf{ref}(\mathbf{I}(A, x, x), \mathbf{ref}(A, x)),\\
\qquad\qquad u)
\end{array}$$

3.2.2 Constancy lemma

On any decidable set a constant function may be defined with values in the same set. The proof will give two families of functions (for an arbitrary set A). Recall that $\mathscr{D}(A)$ stands for (the set representation of) the formula 'A or not A'.

$$\begin{array}{ll}
con_A(z) : A \to A & [z \in \mathscr{D}(A)]\\
iscon_A(z, x, y) \in I(con_A(z, x), con_A(z, y)) & [z \in \mathscr{D}(A); x, y \in A]
\end{array}$$

Assume we have a set A. By $+$-elimination, applied to the constant family of sets A $[z \in \mathscr{D}(A)]$, we define a function $con_A : \mathscr{D}(A) \to A \to A$ subject to the definitional equalities

$$\begin{cases}
con_A(i(x), a) = x & [x \in A]\\
con_A(j(f), a) = a & [f \in (\Pi x \in A)N_0]
\end{cases}$$

and, to define the function $iscon_A$, i.e. to prove that $con_A(d)$ is a constant function for an arbitrary element d of $\mathscr{D}(A)$, we assume that we have two elements a and a' of A, and consider the family of propositions (sets) C given by $C(z) = con_A(z, a) \simeq con_A(z, a')$ $[z \in \mathscr{D}(A)]$.

By $+$-elimination, $C(z)$ is proved for arbitrary $z \in \mathscr{D}(A)$ when we have proved the special cases, $C(i(x))[x \in A]$ and $C(j(f))[f \in (\Pi x \in A)N_0]$. Recalling the definition of C and the equations of *con*, this leaves us with proving, after simplification, in the first case, that $I(x, x)$ is inhabited for $x \in A$, and in the second, that $I(a, a')$ is inhabited for $f \in (\Pi x \in A)N_0$. The first case is solved by reflexivity of identity, and

the second case is solved by absurdity elimination – we can apply f to any of the two elements a and a' of A, to get an element of N_0.

$$
\begin{array}{l}
\mathrm{con} \in (A \in \mathbf{Set};\\
\qquad\quad d \in \mathrm{Dec}(A);\\
\qquad\quad a \in A\\
\qquad\quad)A\\
\qquad \mathrm{con} \equiv\\
\qquad\quad [A, d, a]\\
\qquad\qquad \mathbf{when}(A,\\
\qquad\qquad\qquad \mathbf{Prod}(A, [h]\mathbf{N}_0),\\
\qquad\qquad\qquad [h]A,\\
\qquad\qquad\qquad [x]x,\\
\qquad\qquad\qquad [f]a,\\
\qquad\qquad\qquad d)
\end{array}
$$

$$
\begin{array}{l}
\mathrm{iscon} \in (A \in \mathbf{Set};\\
\qquad\quad d \in \mathrm{Dec}(A);\\
\qquad\quad a, a' \in A\\
\qquad\quad)\mathbf{I}(A, \mathrm{con}(A, d, a), \mathrm{con}(A, d, a'))\\
\qquad \mathrm{iscon} \equiv\\
\qquad\quad [A, d, a, a']\\
\qquad\qquad \mathbf{when}(A,\\
\qquad\qquad\qquad \mathbf{Prod}(A, [h]\mathbf{N}_0),\\
\qquad\qquad\qquad [z]\mathbf{I}(A, \mathrm{con}(A, z, a), \mathrm{con}(A, z, a')),\\
\qquad\qquad\qquad [x]\mathbf{ref}(A, x),\\
\qquad\qquad\qquad [f]\mathbf{case}_0([h]\mathbf{I}(A, a, a'), \mathbf{app}(A, [h]\mathbf{N}_0, f, a)),\\
\qquad\qquad\qquad d)
\end{array}
$$

3.2.3 Collapse lemma

If there is defined, on some set, a constant function with a left inverse, then the set is collapsed.

Suppose we have a set A and a constant function $f : A \to A$ with a left inverse $g : A \to A$. This means that we have functions

$$
\begin{aligned}
&is\_c(x, y) \in I(f(x), f(y)) \; [x, y \in A]\\
&is\_li(x) \in I(g(f(x)), x)) \; [x \in A]
\end{aligned}
$$

representing the assumptions that f is constant and that g is a left inverse of f, respectively. For any two elements a and b of A we construct a proof of $a \simeq b$ by composing proofs of $g(f(b)) \simeq b$, $g(f(a)) \simeq g(f(b))$ and $a \simeq g(f(a))$. These three identities follow, in the first case, from g being a left inverse of f, in the second case, by identity mapping, from f being constant, and in the third case, by symmetry of identity, from g being a left inverse of f. Using the groupoid notation for identity proofs, the proof of $a \simeq b$ is obtained as $is\_li(b) \cdot \widehat{g}(is\_c(a, b)) \cdot (is\_li(a))^{-1}$.

$$
\begin{array}{l}
\text{collaps} \in (A \in \mathbf{Set};\\
\qquad f \in (A)A;\\
\qquad is\_c \in (x, x' \in A)\,\mathbf{I}(A, f(x), f(x'));\\
\qquad g \in (A)A;\\
\qquad is\_li \in (x \in A)\,\mathbf{I}(A, g(f(x)), x);\\
\qquad a, b \in A\\
\qquad)\,\mathbf{I}(A, a, b)\\
\quad \text{collaps} \equiv\\
\qquad [A, f, is\_c, g, is\_li, a, b]\\
\qquad\quad \text{cmpI}(A,\\
\qquad\qquad a,\\
\qquad\qquad g(f(a)),\\
\qquad\qquad b,\\
\qquad\qquad \text{cmpI}(A, g(f(a)), g(f(b)), b, is\_li(b), \text{mapI}(A, A, f(a), f(b), g, is\_c(a, b))),\\
\qquad\qquad \text{invI}(A, g(f(a)), a, is\_li(a)))
\end{array}
$$

3.2.4 *Left inverse lemma*

For any family of functions $v_{xy} : I(x, y) \to I(x, y)\ \ [x, y \in A]$ *there is a corresponding family of left inverses,* $v'_{xy} : I(x, y) \to I(x, y)\ \ [x, y \in A]$.

The proof will give a definition of v'_{xy} together with a family of functions

$$v''_{xy}(z) \in I(v'_{xy}(v_{xy}(z)), z)\ [x, y \in A; z \in I(x, y)].$$

We assume a set A is given with a family of functions $v_{xy} : I(x, y) \to I(x, y)\,[x, y \in A]$ and we will construct a corresponding family of left inverses v'_{xy}. We use the groupoid operations to define v'_{xy} by the equation $v'_{xy}(v) = v \cdot (v_{xx}(r_A(x)))^{-1}\ [v \in I_A(x, y)]$, and, to define v''_{xy}, that is, to prove that v'_{xy} is a left inverse of v_{xy}, we use I-elimination with the family of sets $v'_{xy}(v_{xy}(z)) \simeq z\ [x, y \in A;\ z \in I(x, y)]$. This leaves us, after simplification, with proving for an arbitrary element x of A, that $v_{xx}(r(x)) \cdot (v_{xx}(r(x)))^{-1} \simeq r(x)$. But this is an instance of the right inverse groupoid law of section 3.2.1.

$$
\begin{array}{l}
\text{leftinv} \in (A \in \mathbf{Set};\ nt \in (x, y \in A; \mathbf{I}(A, x, y))\,\mathbf{I}(A, x, y);\ a, b \in A;\ \mathbf{I}(A, a, b))\ \mathbf{I}(A, a, b)\\
\quad \text{leftinv} \equiv [A, nt, a, b, v]\text{cmpI}(A, a, a, b, v, \text{invI}(A, a, a, nt(a, a, \mathbf{ref}(A, a))))\\
\text{isleftinv} \in (A \in \mathbf{Set};\\
\qquad nt \in (a, b \in A; \mathbf{I}(A, a, b))\,\mathbf{I}(A, a, b);\\
\qquad a, b \in A;\\
\qquad u \in \mathbf{I}(A, a, b)\\
\qquad)\,\mathbf{I}(\mathbf{I}(A, a, b), \text{leftinv}(A, nt, a, b, nt(a, b, u)), u)\\
\quad \text{isleftinv} \equiv\\
\qquad [A, nt, a, b, u]\\
\qquad\quad \mathbf{elimI}(A,\\
\qquad\qquad a,\\
\qquad\qquad b,\\
\qquad\qquad [x, y, w]\mathbf{I}(\mathbf{I}(A, x, y), \text{leftinv}(A, nt, x, y, nt(x, y, w)), w),\\
\qquad\qquad [x]\text{invrI}(A, x, x, nt(x, x, \mathbf{ref}(A, x))),\\
\qquad\qquad u)
\end{array}
$$

3.2.5 *DI⊆CI-theorem*

Any set with decidable identity sets has unique identity proofs.

The three preceding lemmas are combined in the following way to yield a proof of the DI⊆CI-theorem.

Assume that A is a set with decidable identities, so that we have a function $d(x, y) \in \mathscr{D}(I(x, y))$ $[x, y \in A]$. Using the constancy lemma, we define a family of constant functions $v_{xy} : I(x, y) \to I(x, y)$ $[x, y \in A]$, any of which, by the left inverse lemma, has a left inverse. Thus we have, on every identity set $I(x, y)$ with $x, y \in A$, a constant function with a left inverse, so we may, by the collapse lemma, conclude that the set $I(x, y)$ is collapsed for arbitrary elements x and y of A.

$$
\begin{array}{l}
\text{condi} \in (A \in \mathbf{Set};\\
\qquad\qquad di \in (x, y \in A)\,\mathrm{Dec}(\mathbf{I}(A, x, y));\\
\qquad\qquad x, y \in A;\\
\qquad\qquad u \in \mathbf{I}(A, x, y)\\
\qquad\qquad)\,\mathbf{I}(A, x, y)\\
\qquad \text{condi} \equiv\\
\qquad\qquad [A, di, x, y, u]\\
\qquad\qquad\qquad \mathrm{con}(\mathbf{I}(A, x, y), di(x, y), u)\\
\text{dici} \in (A \in \mathbf{Set};\\
\qquad di \in (x, y \in A)\,\mathrm{Dec}(\mathbf{I}(A, x, y));\\
\qquad a, b \in A;\\
\qquad u, v \in \mathbf{I}(A, a, b)\\
\qquad)\,\mathbf{I}(\mathbf{I}(A, a, b), u, v)\\
\quad \text{dici} \equiv\\
\qquad [A, di, a, b, u, v]\\
\qquad\quad \mathrm{collaps}(\mathbf{I}(A, a, b),\\
\qquad\qquad\qquad \mathrm{condi}(A, di, a, b),\\
\qquad\qquad\qquad \mathrm{iscon}(\mathbf{I}(A, a, b), di(a, b)),\\
\qquad\qquad\qquad \mathrm{leftinv}(A, \mathrm{condi}(A, di), a, b),\\
\qquad\qquad\qquad \mathrm{isleftinv}(A, \mathrm{condi}(A, di), a, b),\\
\qquad\qquad\qquad u,\\
\qquad\qquad\qquad v)
\end{array}
$$

3.3 ΣDI-theorem

The class of DI-sets is closed under indexed sum.

Our purpose is to show the relevance of the DI⊆CI-theorem and we will be rather brief on other parts of the proof. We assume a set A and a family of sets $B(x)[x \in A]$ fixed in the rest of this section. We use the following notations for the first and second projections.

$$
\begin{aligned}
&p : \Sigma(A, B) \to A\\
&q(z) \in B(p(z))[z \in \Sigma(A, B)]
\end{aligned}
$$

We suppress the proof of the following characterization of identity on dependent pairs up to logical equivalence.

IΣ-lemma. Two elements c and c' of $\Sigma(A, B)$ are identical if and only if there is a proof $u \in I(p(c), p(c'))$ of $p(c) \simeq p(c')$ such that $B^*_{p(c)p(c')}(u, q(c)) \simeq q(c')$.

Next we assume that A and $B(a)$ for arbitrary $a \in A$ both have decidable identities and we will prove that $\Sigma(A, B)$ has decidable identities, that is, we will prove that $c \simeq c'$ is decidable for arbitrary elements c and c' of $\Sigma(A, B)$. By Σ-eliminations we may assume that c and c' are given as pairs, (a, b) and (a', b'), respectively, with a

and a' elements of A and b and b' elements of $B(a)$ and $B(a')$, respectively, and our task is to prove the proposition D that $(a, b) \simeq (a', b')$ is decidable.

Since A has decidable identities, D will follow from proofs of $a \simeq a' \supset D$ and $a \not\simeq a' \supset D$. (We write $a \not\simeq a'$ for $a \simeq a' \supset \bot$.) The second implication is more or less trivial since, by the IΣ-lemma, $(a, b) \simeq (a', b') \supset a \simeq a'$. To prove the first implication, we prove D under the additional assumption of an element u of $I(a, a')$.

Now, since $B(a')$ has decidable identities, D will follow from proofs of $B^*_{aa'}(u, b) \simeq b' \supset D$ and $B^*_{aa'}(u, b) \not\simeq b' \supset D$. The first implication is more or less trivial, since, by the IΣ-lemma, the antecedent implies $(a, b) \simeq (a', b')$. To prove the second implication, we will prove $(a, b) \not\simeq (a', b')$ under the additional assumption of $B^*_{aa'}(u, b) \not\simeq b'$. This, in turn, we will prove by deriving an absurdity from the additional assumption $(a, b) \simeq (a', b')$. By the IΣ-lemma (and Σ-elimination), absurdity will follow from a proof of absurdity under the additional assumption of an element u' of $I(a, a')$ such that $B^*_{aa'}(u', b) \simeq b'$.

Summing up the interesting assumptions available to us in our quest of the absurdity proof that will close the proof of the ΣDI-theorem,

$$\begin{array}{ll} u \in I(a, a') & B^*_{aa'}(u, b) \not\simeq b' \\ u' \in I(a, a') & B^*_{aa'}(u', b) \simeq b', \end{array}$$

the relevance of the DI$\subseteq$CI-theorem becomes apparent, since this will, applied to the assumption that A has decidable identities, show that the identity proofs u and u' are identical – then, by identity coercion, $B^*_{aa'}(u, b) \not\simeq b$ implies $B^*_{aa'}(u', b) \not\simeq b$ which in combination with $B^*_{aa'}(u', b) \simeq b'$ is absurd.

4 DI$\subseteq$CI in the context of conventional equality

We give two coherence theorems in type theory without inductive equality. The first theorem purifies the category theoretical view on identity and was obtained as an adaptation of the DI$\subseteq$CI-theorem to work on categorical identity instead of inductive equality. The second theorem reduces the problem of finding an inital category structure to that of finding a decidable logical identity.

4.1 Coherence theorem 1

If a set has an initial category structure with decidable arrow sets, then these are collapsed with respect to arrow equality.

Before the proof we observe, going back to the context of inductive equality, that, after constructing the identity category and proving its initial property, coherence theorem 1 can immediately be applied to a decidable inductive equality and give a proof of the DI$\subseteq$CI-theorem.

We begin the proof with some lemmas about categorical identity. These will be developed under the assumption of a fixed set A with a fixed initial category structure with arrow sets $I(a, b)$, the usual notation for the arrow operations and arrow equality $\overset{xy}{\simeq}$. We will then add the assumption about decidability and show

how the proof given in section 3 of the DI⊆CI-theorem may be modified to give a proof of coherence theorem 1.

4.1.1 Identity induction

Using the indexed sum and the existence and uniqueness of category structure homomorphisms defined on the initial category structure, we may establish the following proof principle.

If $P_{xy}[x, y \in A]$ is a family of arrow predicates[7] which holds for the unit arrows, is stable under arrow composition and is compatible with arrow equality, and if a and b are elements of A, then P_{ab} holds for every arrow in $I(a, b)$.

Using the assumptions about P, we can define a new category structure on A, with arrow sets $I'(a, b) = (\Sigma u \in I(a, b))P_{ab}(u)$. Two arrows in $I'(a, b)$ are considered as equal when their first components are equal in the sense of $\overset{ab}{\simeq}$. The rest of the new category is done so that the first projection $fst_{ab} : I'(a, b) \to I(a, b)$ becomes a functor. The initiality of I gives a right inverse $\delta_{ab} : I(a, b) \to I'(a, b)$ to the first projection fst_{ab}. The meaning of this is that $fst_{ab}(\delta_{ab}(u)) \overset{ab}{\simeq} u$ for an arbitrary arrow $u \in I(a, b)$. The second component of $\delta_{ab}(u)$ is a proof that the first component $fst_{ab}(\delta_{ab}(u))$ has the property P_{ab}, so we may, since P is compatible with arrow equality, conclude that u also has the property P_{ab}.

4.1.2 The initial category is a groupoid

Since the dual of the initial category also is a category, initiality gives a contravariant functor sending the arrow $u \in I(a, b)$ to an arrow $u^{-1} \in I(b, a)$. To prove the left and right inverse laws, showing that u^{-1} is an inverse arrow of u, we may apply identity induction to the arrow predicates $P_{ab}(u) = u^{-1} \cdot u \overset{ab}{\simeq} 1$ and $P'_{ab}(u) = u \cdot u^{-1} \overset{ab}{\simeq} 1$, respectively.

4.1.3 Left inverse lemma

Although we could prove a naturality lemma of the general form indicated in section 3.1.1, we will consider the special case of a family of arrow functions

$$v_{xy} : I(x, y) \to I(x, y)[x, y \in A]$$

which is assumed to respect arrow equality, and show that

$$v_{ay}(u \cdot v) \overset{ay}{\simeq} u \cdot v_{ax}(v)[a, x, y \in A; u \in I(x, y); v \in I(a, x)],$$

which we may do by identity induction on the arrow predicate

$$P_{xy}(u) = (\forall w \in I(a, x))\, v_{ay}(u \cdot w) \overset{ay}{\simeq} u \cdot v_{ax}(w),$$

[7] More precisely, $P_{xy}(u)$ is a set for $x, y \in A$ and $u \in I(x, y)$.

where a is a fixed element of A. We may then choose $x = a$, $v = 1_a$ and $y = b$ to infer that $v_{ab}(u) \overset{ab}{\simeq} v_{ab}(u \cdot 1_a) \overset{ab}{\simeq} u \cdot v_{aa}(1_a)$, from which we see that v_{ab} has the left inverse v'_{ab} given by

$$v'_{ab}(v) = v \cdot (v_{aa}(1_a))^{-1} \; [v \in I(a,b)].$$

Note that the left inverse v'_{ab} preserves arrow equality because arrow composition does so.

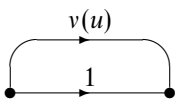
4.1.4 Adding decidability

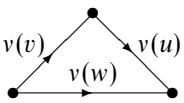
After assuming the arrow sets $I(x,y)$ to be decidable, the proof is as in the case of inductive equality, but with some modifications. First, we here have arrow equality instead of inductive equality on proofs of inductive equality. As a consequence the second part of the constancy lemma of section 3.2.2 is rephrased thus: Instead of proving (for arbitrary elements x, y of A and z of $\mathscr{D}(A)$) that $con_A(z,x)$ and $con_A(z,y)$ are identical in the sense of inductive equality, we prove that they are related by any reflexive relation. Secondly, the collapse lemma is rephrased so that the left inverse is assumed to preserve equality.

The assumption of decidable arrow sets $I(a,b)[a,b \in A]$ now gives a family of $\overset{ab}{\simeq}$-constant functions $v_{ab} : I(a,b) \to I(a,b)$, to which, since a constant function trivially respects equality, the left inverse lemma can be applied to give a family of equality preserving left inverses so that, finally, the collapse lemma can be put to work showing the arrow sets $I(a,b)[a,b \in A]$ to be collapsed (with respect to arrow equality).

4.2 Coherence theorem 2

If a set A has a decidable smallest reflexive relation I with proof sets written $I(x,y)[x,y \in A]$, then I is transitive and if $x \longrightarrow y\,[x,y \in A]$ are the arrow sets in a categorical structure on A, then there is a family of arrow valued functions $v_{xy} : I(x,y) \to (x \longrightarrow y)\,[x,y \in A]$ such that any diagram of one of the following two forms is commutative:





This is a more explicit statement than the one given in the introduction. We will, however, not elaborate on the equivalence of the two formulations.

Remark. The commutative diagrams show that v is functorial (for any definition of the unit and composition operations on identity proofs), and that v_{xy} is a constant function. In fact, if $u, v \in I(x,y)$, then with $r(x) \in I(x,x)$ for v, the commutative triangle shows that $v(w)$ and $v(u) \cdot v(r(x))$ are equal arrows for any $w \in I(x,y)$, in particular for $w = u$ and for $w = v$, showing $v(u)$ and $v(v)$ to both equal $v(u) \cdot v(r(x))$. (Here, as usual, $\cdot$ denotes arrow composition.)

4.2.1 Coherence theorem 2 and DI⊆CI

Before the proof we indicate how the theorem can be used to prove the DI⊆CI-theorem. Clearly, inductive equality is a smallest reflexive relation, so if it is decidable, it can act as I in the theorem. Then, after the construction of the identity category, the theorem gives us a family of functions $v_{xy} : I(x, y) \to I(x, y)$. By the remark above, these are constant with respect to identity. By the commutativity of the left most diagram with $u = r(x)$ we see that any reflexivity proof is mapped to itself. Thus, we may by I-elimination conclude that any identity proof is mapped to itself, so the identity function on $I(x, y)$ is constant and hence $I(x, y)$ is collapsed.

4.2.2 Proof of coherence theorem 2

We assume a set A with a decidable logical identity I (i.e. I is reflexive and is contained in any reflexive relation – we refer to the latter property as the *smallness* of I) with identity proof sets $I(x, y)[x, y \in A]$ and reflexivity proofs $r(x) \in I(x, x)[x \in A]$. We also assume a category structure on A with, as usual, arrow sets written $x \longrightarrow y$, arrow operations 1_x and $f \cdot g$ and arrow equality $\overset{xy}{\simeq}$.

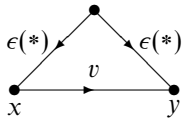
Constant endo functions on the identity proof sets. Just as in the proof of coherence theorem 1, we use decidability to define a family of functions $\gamma_{xy} : I(x, y) \to I(x, y)$ which are constant in the sense that $\gamma_{xy}(u)$ and $\gamma_{xy}(v)$ are related by any reflexive relation on $I(x, y)$.

Epimorphisms and ϵ-arrows. Since the unit arrow clearly is an epimorphism, the relation on A, given by $\mathscr{R}_\epsilon(x, y)$ if and only if there is an epimorphism in $x \longrightarrow y$, is reflexive. Hence, the smallness of I gives a function that maps an arbitrary identity proof $u \in I(x, y)$ to an epimorphism $\epsilon'_{xy}(u) \in x \longrightarrow y$. Composing with γ we define $\epsilon_{xy}(u) = \epsilon'_{xy}(\gamma_{xy}(u))$. Let us introduce the term *ϵ-arrow* for any arrow of the form $\epsilon_{xy}(u)$ with $u \in I(x, y)$. The first of the following properties of ϵ-arrows is immediate from the definition.

1. Any ϵ-arrow is an epimorphism.
2. Any two ϵ-arrows $x \longrightarrow y$ are equal arrows.

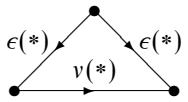
The second property follows from the constancy of γ, since the lifted relation ${\epsilon'_{xy}}^{-1}(\overset{xy}{\simeq})$ clearly is a reflexive relation on $I(x, y)$.

Definition of $v_{xy}(u) : x \longrightarrow y$. Consider the relation $\mathscr{R}_v(x, y)[x, y \in A]$ defined as the existence of an arrow $v : x \longrightarrow y$ such that any diagram the following form is commutative:



The relation $\mathscr{R}_v$ is reflexive since, when $x = y$, $v = 1_x$ has the desired property in virtue the second property of ϵ-arrows (and the left unit category law). The

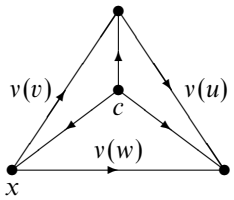
smallness of I then gives a function that maps an arbitrary identity proof $u \in I(x, y)$ to an arrow $v_{xy}(u) : x \longrightarrow y$ in such way that any diagram of the following form is commutative:



We refer to this diagram as the *ϵv-triangle*.

Identity is transitive. Transitivity follows from identity being contained in the obviously reflexive relation $\mathscr{R}(y, z) = (\forall x \in A)(I(x, y) \supset I(x, z))$. We use $u \cdot v$ to denote the element in $I(x, z)$ obtained by transitivity from $u \in I(y, z)$ and $v \in I(x, y)$.

Proof that v has the desired properties



Considering the diagram above, we first note that if the small triangles are commutative and the arrow $c \longrightarrow x$ is an epimorphism, then the large triangle is commutative. So, to prove that the large triangle is commutative, it is sufficient to construct the centre point c and ϵ-arrows outwards, because then the small triangles are of the form indicated in the ϵv-triangle and are thus commutative, and the arrow $c \longrightarrow x$ is an epimorphism because any ϵ-arrow is.

We chose $c = x$ as centre point and the outgoing ϵ-arrows correspond to the identity proofs $r(x)$, $v \cdot r(x)$ and $u \cdot v \cdot r(x)$, respectively, in the clockwise order of the figure.

To prove that any arrow of the form $v_{xx}(u)$ with $u \in I(x, x)$ is an identity arrow, we observe that, by the commutativity of the ϵv-triangle, $v(u) \cdot \epsilon(r(x)) \overset{xx}{\simeq} \epsilon(r(x))$ which in combination with the left unit law and the ϵ-arrows being epimorphisms shows that $v(u) \overset{xx}{\simeq} 1_x$.

Acknowledgements

I thank Peter Dybjer for valuable discussion and for suggesting many improvements of the presentation, Jan Smith for suggesting the translation of the DI⊆CI-theorem to a context of conventional equality and Thomas Schreiber for pointing out the relevance of the DI⊆CI-theorem for reasoning about updating a dependently typed store and for showing me the LEGO version of the DI⊆CI-theorem.

References

Altenkirch, T., Gaspes, V., Nordström, B. and von Sydow, B. (1994) *A User's Guide to ALF*. Chalmers University of Technology, Sweden. (Available on the WWW `file://ftp.cs.chalmers.se/pub/users/alti/alf.ps.Z`.)

Augustsson, L., Coquand, T. and Nordström, B. (1992) A short description of another logical framework. In: G. Huet and G. Plotkin, editors, *Proceedings of the 1st Workshop on Logical Frameworks*, Antibes, France.

Beylin, I. and Dybjer, P. (1996) Extracting a proof of coherence for monoidal categories from a proof of normalization for monoids. In: S. Berardi and M. Coppo, editors, *TYPES '95: Lecture Notes in Computer Science 1158*, pp. 47–61.

Bishop, E. (1967) *Foundations of Constructive Analysis*. McGraw-Hill.

Cartmell, J. (1986) Generalised Algebraic Theories and Contextual Categories. *Annals of Pure and Applied Logic*, **32**, 209–243.

Dybjer, P. (1991) Inductive sets and families in Martin-Löf's type theory and their set-theoretic semantics. *Logical Frameworks*, pp. 280–306. Cambridge University Press.

Dybjer, P. (1997) A general formulation of simultaneous inductive-recursive definition in type theory. *Journal of Symbolic Logic* (to appear).

Greenleaf, N. (1980) Formalizing constructive mathematics: Why and how? In: A. Dold and B. Eckmann, editors, *Constructive Mathematics*, pp. 213–240. Springer-Verlag.

Hofmann, M. (1995) Conservativity of equality reflection over intensional type theory. *Proceedings of the BRA TYPES workshop*, Torino, Italy. Springer-Verlag.

Hofmann, M. (1995) Extensional Concepts in Type Theory. *PhD thesis*, University of Edinburgh.

Hofmann, M. and Streicher, T. (1994) A groupoid model refutes uniqueness of identity proofs. *Proceedings of the 9th Symposium on Logic in Computer Science (LICS)*, Paris, France.

Magnusson, L. and Nordström, B. (1994) The ALF proof editor and its proof engine. *Types for Proofs and Programs: Lecture Notes in Computer Science*, pp. 213–237. Springer-Verlag.

Martin-Löf, P. (1972) An Intuitionistic Theory of Types. *Technical report*, University of Stockholm.

Nordström, B., Petersson, K. and Smith, J. M. (1990) *Programming in Martin-Löf's Type Theory. An Introduction*. Oxford University Press.