Finitary Partial Inductive Definitions as a General Logic

Lars-Henrik Eriksson

Swedish Institute of Computer Science
Box 1263, S-164 28 KISTA, SWEDEN
E-mail: lhe@sics.se

Abstract. We describe how the calculus of partial inductive definitions is used to represent logics. This calculus includes the powerful principle of definitional reflection. We describe two conceptually different approaches to representing a logic, both making essential use of definitional reflection. In the deductive approach, the logic is defined by its inference rules. Only the succedent rules (in a sequent calculus setting – introduction rules in a natural deduction setting) need be given. The other rules are obtained implicitly using definitional reflection. In the semantic approach, the logic is defined using its valuation function. The latter approach often provides a more straightforward representation of logics with simple semantics but complicated proof systems.

1 Introduction: Finitary Partial Inductive Definitions

We will describe how to use the calculus of *partial inductive definitions* as a general logic. That is, as a framework for representing various logics. Following common practise, we will refer to the calculus of partial inductive definitions as the *metalogic* and call a logic being represented an *object logic*. To make the paper self-contained, we will begin with a summary of the calculus of partial inductive definitions. Actually, we will describe a finitary version of it since the proper calculus is infinitary, and thus unsuitable for use as a metalogic. The finitary version is described in detail in [6], and the original infinitary theory in [12]. The presentation given here is slightly different from that of [6].

Formulae of the metalogic (called *conditions*) will be the following:

t	where t is an expression of the simply typed lambda calculus (an *atomic condition*)
$C_1,C_2,\ldots,C_n$	where every C_i is a condition, $n\geq 0$ (*conjunction*).
$C \Rightarrow C'$	where C and C' are conditions (*implication*).
$\Pi x\, C(x)$	where C is a condition containing the variable x. x is bound by the operator Π (*universal quantification*).

Although the inference rules for the metalogical implication and quantification will be similar to the corresponding logical connectives, they are not implication and quantification in the same sense as in logic.

Parentheses will be used when necessary. $A,B \Rightarrow C,D$ will be taken as $A,(B \Rightarrow C),D$.

It is orthogonal to the rest of the calculus what kind of expressions the atomic conditions really are, as long as they have a decidable equality and a standard notion of substitution. For our purposes, simply typed lambda expressions are suitable. We will deliberately be vague about the particular types of lambda expressions treating them almost as untyped expressions. For our purposes, the typing mechanism primarily serves as a clerical aid and to prevent the formation of non-normalisable expressions.

The formulation of simply typed lambda expressions that we will use differs from the usual formulation in that there are two kinds of free variables. One kind is called *parameters*, the other will be called *ordinary variables*. This distinction is only important for free variables – we will regard bound variables as ordinary variables. Free ordinary variables will be written A, B, C,... Parameters will be written A^*, B^*, C^*... Substitutions operating specifically on parameters will be written σ^*, τ^*, etc.

In the sequel, we will use the term "variables" solely to refer to ordinary variables. In particular, *ground terms* refer to terms without free variables, but which may contain parameters. For the purposes of this paper, the distinction between parameters and ordinary variables is not essential, but we uphold the distinction to be consistent with the presentation in [6].

A *clause* is an expression of the form $H \Leftarrow B$ where the *head* H is an atomic condition and the *body* B is an arbitrary condition. When the body is the empty conjunction, it is omitted entirely. Clauses may contain free variables, which should be regarded as being universally quantified over the clause. In the sequel, we will assume that each time a clause is used, any variables in the clause are renamed as required to avoid conflicts with other variables.

A *partial inductive definition* (or just a *definition*) is a finite set of clauses. The instances of the atomic conditions in the heads of the clauses are said to be *defined by* the definition. We will adopt the convention that for every definition P, there is some atomic condition, written $\bot$, that is not defined by P.

The *parameter transform* of a definition is obtained by substituting parameters for all free variables in the definition; different parameters are used for different variables. As with variables, we assume that parameters are renamed as necessary to avoid conflicts.

The calculus of partial inductive definitions is a sequent calculus system, where the formulae occurring in sequents are conditions. The exact form of some of the inference rules depends on a particular definition P. To highlight this dependency the turnstile symbol is often indexed with the particular P, i.e. $\vdash_P$. To simplify the presentation of the inference rules, we will assume that antecedents of sequents are unordered multisets.

The deductive system is built up from the following axiom schema and inference rules.

$$\frac{\Gamma, C, C \vdash C'}{\Gamma, C \vdash C'} \text{ contraction} \qquad \frac{\Gamma \vdash C'}{\Gamma, C \vdash C'} \text{ weakening}$$

$$\frac{\Gamma \vdash C_1 \quad \cdots \quad \Gamma \vdash C_n}{\Gamma \vdash (C_1, \ldots, C_n)} \vdash() \qquad \frac{\Gamma, C_1, \ldots, C_n \vdash C}{\Gamma, (C_1, \ldots, C_n) \vdash C} ()\vdash$$

$$\frac{\Gamma, C \vdash C'}{\Gamma \vdash C \Rightarrow C'} \vdash\Rightarrow \qquad \frac{\Gamma \vdash C' \quad \Gamma, C'' \vdash C}{\Gamma, C' \Rightarrow C'' \vdash C} \Rightarrow\vdash$$

$$\frac{\Gamma \vdash C(X^*)}{\Gamma \vdash \Pi x\, C(x)} \vdash\Pi \qquad \frac{\Gamma, C(t) \vdash C'}{\Gamma, \Pi x\, C(x) \vdash C'} \Pi\vdash$$

where X^* has the same type as x and does not occur in Γ or C.

where t is some arbitrary ground formula of the same type as x.

$\Gamma,a\vdash a$ axiom

$$\frac{\Gamma\vdash B\sigma}{\Gamma\vdash a}\ \vdash\mathrm{D}$$

where $H\Leftarrow B$ is a clause in P, such that $a=H\sigma$ and $B\sigma$ is ground.

$$\frac{\Gamma\sigma_1^*,B_1\sigma_1^*\vdash C\sigma_1^*\ \cdots\ \Gamma\sigma_n^*,B_n\sigma_n^*\vdash C\sigma_n^*}{\Gamma,a\vdash C}\ \mathrm{D}\vdash$$

where $H_i \Leftarrow B_i$, $1\le i\le n$, are exactly those clauses in the parameter transform of P where the heads unify with a. Each σ_i^* is the corresponding most general unifier, e.g. $\sigma_i^*=mgu(a,H_i)$. The clauses must not have any parameters in common with Γ, a or C.

The D$\vdash$ rule expresses the principle of *definitional reflection* [20]. An important special case of this rule is when no clause heads unify with a. In that case n=0 so the inference step has no premises. That is, the conclusion sequent is immediately proved. We say that a is absurd, and that the conclusion sequent is proved by contradiction. The purpose of the convention of always having the undefined expression $\perp$ is to always have something that is absurd.

This formulation of the D$\vdash$ rule presupposes that most general unifiers (mgus) exist whenever two expressions are unifiable. That is not always the case with higher-order expressions. In this paper, all higher-order expressions will be so-called *higher-order patterns* [16] where mgus always exist. In the more general case, there must be one premise for each clause and each unifier in some complete set of unifiers for a and the head of that clause – see [6] for details.

An important property of this calculus is that the cut inference rule is not in general admissible. For some classes of definitions, including cut as an inference rule strictly increases the set of derivable sequents. That is, for some P, cut-elimination is not possible. P is said to be *total* iff cut is an admissible rule. In section 4, we give a definition for which any sequent would be derivable using cut, but where the set of derivable sequents is still interesting when cut is not used.

If either $\vdash_P a$ or $a\vdash_P \perp$, for any parameter-free atomic condition a, then P is called a *complete* definition.

All inference rules and the axiom schema are closed under substitution of parameters. It is occasionally useful to make this explicit by using the following admissible rule:

$$\frac{\Gamma\vdash C}{\Gamma\sigma^*\vdash C\sigma^*}\ \text{specialisation}$$

To simplify the presentation of a derivation, we will always apply the $(\ )\vdash$ or the $\vdash(\ )$ rules without explicit mention whenever a sequence condition appears. E.g. if the definition contains a clause $a \Leftarrow b,c$ we will write:

$$\frac{\Gamma\vdash b\qquad\Gamma\vdash c}{\Gamma\vdash a}\ \vdash\mathrm{D}\quad\text{rather than}\quad\frac{\dfrac{\Gamma\vdash b\qquad\Gamma\vdash c}{\Gamma\vdash(b,c)}\ \vdash()}{\Gamma\vdash a}\ \vdash\mathrm{D}$$

2 Representing logics

How can the theory of partial inductive definitions be used to represent a logic? The starting point is the fact that two of the inference rules of the metalogic – the $\vdash$D and D$\vdash$ rules – are dependent on a particular definition for their exact form. These rules can be thought of not as two ordinary inference rules making reference to the definition, but as two inference rule schemata, generating a set of antecedent and succedent rules depending on the particular definition. The $\vdash$D and D$\vdash$ rules can be said to be "instantiated" by the definition. For example, given the partial inductive definition FOL from the next section with the two clauses $A \vee B \Leftarrow A$ and $A \vee B \Leftarrow B$ intended to define object-level disjunction ($\vee$), the instances obtained are

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \vee B}\ \vdash\mathrm{D} \qquad \frac{\Gamma \vdash B}{\Gamma \vdash A \vee B}\ \vdash\mathrm{D} \qquad\qquad \frac{\Gamma, A \vdash C \qquad \Gamma, B \vdash C}{\Gamma, A \vee B \vdash C}\ \mathrm{D}\vdash$$

In the same way, one or several $\vdash$D instance is obtained for each clause, and one D$\vdash$ instance for each set of clauses with non-unifiable heads. The similarities between these instances and inference rules of the first-order sequent calculus are no coincidence. This view of the calculi of partial inductive definitions as being "instantiable" to other systems with a specialised set of inference rules is the main motivation for using the calculus of finitary partial inductive definitions as a general logic.

Expressions of the object logic will be represented by expressions of typed lambda calculus in the standard way (see [14] for a detailed presentation in the context of the Edinburgh Logical Framework). Free and bound variables of the metalogic will typically be used to represent free and bound variables of the object logic. We will generally let the expressions of the object logic themselves be used as syntactic sugar for their representation. E.g. we will generally write $\forall x.p(x)$ instead of its representation $\forall(\lambda x.p(x))$. Atomic formulae of the object logic – i.e. formulae that are not built up using logical constants – will be represented using a "tag" such as g. A formula such as $p(x) \wedge q$ would be represented as $g(p(x)) \wedge g(q)$. The purpose of this tag is to simulate a subtype scheme, so that it is possible to give a pattern unifying exactly with the atomic formulae of the object logic – e.g. $g(X)$. Again, in the interests of readability, the formula $p(x) \wedge q$ will be used as syntactic sugar for its own representation.

When discussing the representation of a logic, we will use the term *judgment*. A judgment is the unit of reasoning of an object logic. Different kinds of judgments are called *judgment forms*. In predicate logic, we would have a judgment form expressing that a particular formula is true. Other formal systems could use different judgments forms, e.g. a type theory could have one judgment form to express that an object is a type and another judgment form to express that an object is of a certain type. Since the judgment form *true A*, stating that A is true is so common, we will generally identify a judgment *true A* with A itself to reduce the amount of writing.

It is also possible that for technical reasons the representation of a logic introduces new judgment forms that have no direct counterpart in the logic itself. Complicated side conditions on inference rules could be expressed in this way, e.g. a representation of a modal logic could have a judgment form stating that a formula is comodal.

We will describe two conceptually different ways to represent a logic using partial inductive definitions. In the first approach, which we will call the *deductive* approach, the emphasis is on the *derivability* of a judgment. The inference rules of the logic are encoded

as a partial inductive definition. That a judgment holds according to the definition is interpreted as it being derivable in the object logic. The metalogic will essentially be turned into a sequent calculus system for the object logic.

The main difference of the deductive approach from other work on general logic is that only the succedent rules of the sequent calculus (corresponding to the introduction rules in a natural deduction setting) need be given. The antecedent (elimination) rules are obtained implicitly using the principle of definitional reflection (D $\vdash$ rule). This use of definitional reflection is unique among general logics, although it has recently been applied to Martin-Löf's type theory [3].

The other approach will be called the *semantic* approach. Here, the emphasis is on representing the *truth* of a judgment in an *interpretation* according to some semantics. The semantics of the logic is encoded as a partial inductive definition. That a judgment holds according to that definition is interpreted as it being true according to the interpretation. Given the definition, the metalogic will behave as a sequent calculus system to deduce truth and falsity in interpretations. That a sequent holds implies that the succedent judgment is true or at least one judgment in the antecedent is false.

It is also possible to use the semantic approach to represent truth in all interpretations, i.e. logical truth. We obtain a specialised calculus with inference rules appropriate for the logic in question, sharing the same general properties, such as structural rules, with the calculus of the metalogic. In a sense, the semantic approach *embeds* the object logic into the metalogic. Of course, the inference rules may or may not correspond to the inference rules of deductive characterisations of that logic, depending on how the semantic definition is written.

The semantic approach is, perhaps, the most important contribution of this work. The deductive approach is included mainly to show that we can represent a logic using inference rules. The semantic approach is the more interesting one.

Although the two approaches are different in motivation, they lead to very much the same representation of the logic when they are both applicable (e.g. the same representation of predicate logic will be motivated using both approaches). This should not be surprising, considering the close relationship in practise between the notions of truth and derivability.

3 The Deductive Approach

When defining a logic using the deductive approach, we write down a description of the succedent rules of the logic (corresponding to the introduction rules in a natural deduction setting) as the clauses of a partial inductive definition. In such a clause, the nonatomic conditions (conjunction, implication and quantification) are used to express the desired form of the succedent rule. Each clause will provide a particular instance of the $\vdash$D rule which, together with the other inference rules of the metalogic, will generate the desired inference rule of the object logic. E.g., the clause $A{\rightarrow}B \Leftarrow A{\Rightarrow}B$, states that an implication in the succedent can be derived from a sequent with B as succedent and A in the antecedent, i.e. the rule

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A{\rightarrow}B}\ \vdash{\rightarrow}$$

which is the rule for implication in the antecedent in Gentzen's system LJ for intuitionistic first-order predicate logic [9]. Instantiating the $\vdash$D rule with the clause and using the $\vdash\Rightarrow$ rule to derive the premise of the $\vdash$D rule, we get the derivation schema:

$$\dfrac{\dfrac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}\ \vdash\Rightarrow}{\Gamma \vdash A \rightarrow B}\ \vdash\mathrm{D}$$

which admits the same inferences as the rule from LJ. Antecedent rules (corresponding to elimination rules in a natural deduction setting) are obtained using the rule of definitional reflection – D$\vdash$. Instantiating the D$\vdash$ rule with the sample clause and using the $\Rightarrow\vdash$ rule to derive the premise of the D$\vdash$ rule, we get the following derivation schema:

$$\dfrac{\dfrac{\Gamma, B \vdash C \qquad \Gamma \vdash A}{\Gamma, A \Rightarrow B \vdash C}\ \Rightarrow\vdash}{\Gamma, A \rightarrow B \vdash C}\ \mathrm{D}\vdash$$

which admits the same inferences as the rule for implication in the antecedent in LJ.

$$\dfrac{\Gamma \vdash A \quad \Gamma, B \vdash C}{\Gamma, A \rightarrow B \vdash C}\ \rightarrow\vdash$$

(Actually, this is not the exact rule from LJ, as that rule has different Γ in each premise and the conclusion has both of them. Since LJ includes rules for contraction and weakening, the formulation here is equivalent to that of LJ.)

Before giving complete examples of logic representations, we turn to the question of what logics we can represent using the deductive approach.

Apart from the clause-based succedent rule ($\vdash$D), the principle of definitional reflection (D$\vdash$), and inference rules for the nonatomic conditions, the sequent calculus includes the structural inferences of contraction, weakening and permutation. The first two are proper inference rules, the last one is implicit in viewing the antecedent of a sequent as a multiset. A sequent calculus with these features is called a *structural framework* by Schroeder-Heister [19].

This puts some constraints on what logics we can represent. Any logics that are represented directly must admit general contraction, weakening and permutation. Logics that do not – so called "substructural logics" (e.g. linear logic, relevance logic and Lambek calculus) can not be represented – at least not in a straightforward way. In [19], Schroeder-Heister argues that structural and logical features of deductive systems can be separated in a natural way. (Schroeder-Heister does not consider logics with quantification, but that generalisation is straightforward.) The structural features are given by the structural framework, while the logical features are given by a database of clauses defining the succedent rules. Following this view, we could represent a logic by giving the structural framework in addition to the database of clauses. Schroeder-Heister shows how structural frameworks different from that of the partial inductive definitions can be used to express substructural logics in a natural way.

We only admit sequents with a single condition in the succedent. This makes it less convenient – but not impossible – to define classical logics, such as Gentzen's system LK for classical first-order predicate calculus. Although not discussed by Schroeder-Heister, the number of elements of the succedent (one or several) would be a natural feature to be determined by the structural framework.

Since all inference rules of the object logic are obtained using the inference rules of the metalogic, the only side conditions on inference rules that can be directly represented are side conditions that have a counterpart in the metalogic. There is, in fact, only one such side condition, namely the restrictions on parameters in some of the rules. Any other side conditions must be expressed indirectly by introducing additional judgments for those side conditions and adding clauses to define the inference rules for those judgments.

A final constraint on the logics we can represent is the connection between the antecedent and succedent rules of the object logic given by the principle of definitional reflection – there is no independence of the antecedent and succedent rules. This strongly limits any attempts to use the rules for other purposes than giving meaning to logical constants. Suppose that we carelessly tried to define the principle of reductium ad absurdum by writing a clause

$$A \Leftarrow \neg\neg A$$

where A is an arbitrary formula. This would give the desired succedent rule, but would also have the undesired effect of adding an extra premise to every antecedent rule of the object logic, e.g. the D$\vdash$ instance for $A \rightarrow B$, would be the useless rule

$$\frac{\Gamma, A \Rightarrow B \vdash C \quad \Gamma, \neg\neg(A \rightarrow B) \vdash C}{\Gamma, A \rightarrow B \vdash C}\ \mathrm{D}\vdash$$

To summarise, we can *directly* represent logics that admit the full set of structural rules and have sequents with a singleton succedent, only the parameter side condition or side conditions that can be expressed as judgments, and finally the connection between antecedent and succedent rules given by definitional reflection.

These constraints may seem rather severe, but it should be kept in mind that they apply to the direct representation of a logic. By introducing additional judgments, they can often be worked around. Also, they are not unique to our framework. E.g. the Edinburgh Logical Framework (LF) [14] and the metalogic of Isabelle [18] share – apart from the lack of connection between antecedent and succedent rules – the same basic constraints. Still, many nontrivial logics can be represented in these frameworks – possibly using extra machinery to overcome the constraints (see e.g. [1] for the case of LF).

The following partial inductive definition (which we will call FOL) defines first-order (intuitionistic) logic.

$$A \wedge B \Leftarrow A, B$$
$$A \vee B \Leftarrow A$$
$$A \vee B \Leftarrow B$$
$$A \rightarrow B \Leftarrow A \Rightarrow B$$
$$\neg A \Leftarrow A \Rightarrow \bot$$
$$\forall x A(x) \Leftarrow \Pi x A(x)$$
$$\exists x A(x) \Leftarrow A(X)$$

Here the variables X and x have the type of terms of the object logic, A and B have the type of formulae (or functions from terms to formulae), and P has the type of atomic formulae – except equalities.

Instantiating the $\vdash$D and D$\vdash$ rules with these clauses, and using the other rules of the metalogic to derive premises containing non-atomic conditions, we get the following derivation schemes:

$$\frac{\Gamma\vdash A \qquad \Gamma\vdash B}{\Gamma\vdash A\wedge B}\ \vdash\mathrm{D} \qquad\qquad \frac{\Gamma, A, B\vdash C}{\Gamma, A\wedge B\vdash C}\ \mathrm{D}\vdash$$

$$\frac{\Gamma\vdash A}{\Gamma\vdash A\vee B}\ \vdash\mathrm{D} \qquad \frac{\Gamma\vdash B}{\Gamma\vdash A\vee B}\ \vdash\mathrm{D} \qquad\qquad \frac{\Gamma, A\vdash C \qquad \Gamma, B\vdash C}{\Gamma, A\vee B\vdash C}\ \mathrm{D}\vdash$$

$$\frac{\dfrac{\Gamma, A\vdash B}{\Gamma\vdash A\Rightarrow B}\ \vdash\Rightarrow}{\Gamma\vdash A\to B}\ \vdash\mathrm{D} \qquad\qquad \frac{\dfrac{\Gamma, B\vdash C \qquad \Gamma\vdash A}{\Gamma, A\Rightarrow B\vdash C}\ \Rightarrow\vdash}{\Gamma, A\to B\vdash C}\ \mathrm{D}\vdash$$

$$\frac{\dfrac{\Gamma, A\vdash \bot}{\Gamma\vdash A\Rightarrow \bot}\ \vdash\Rightarrow}{\Gamma\vdash \neg A}\ \vdash\mathrm{D} \qquad\qquad \frac{\dfrac{\dfrac{}{\Gamma, \bot\vdash C}\ \mathrm{D}\vdash \qquad \Gamma\vdash A}{\Gamma, A\Rightarrow \bot\vdash C}\ \Rightarrow\vdash}{\Gamma, \neg A\vdash C}\ \mathrm{D}\vdash$$

$$\frac{\dfrac{\Gamma\vdash A(X^*)}{\Gamma\vdash \Pi x\, A(x)}\ \vdash\Pi}{\Gamma\vdash \forall x\, A(x)}\ \vdash\mathrm{D} \qquad\qquad \frac{\dfrac{\Gamma, A(t)\vdash C}{\Gamma, \Pi x\, A(x)\vdash C}\ \Pi\vdash}{\Gamma, \forall x\, A(x)\vdash C}\ \mathrm{D}\vdash$$

$$\frac{\Gamma\vdash A(t)}{\Gamma\vdash \exists x\, A(x)}\ \vdash\mathrm{D} \qquad\qquad \frac{\Gamma, A(X^*)\vdash C}{\Gamma, \exists x\, A(x)\vdash C}\ \mathrm{D}\vdash$$

These derivation schemes correspond almost exactly to the inference rules of Gentzen's system LJ. There is an insignificant difference in the rule for implication in the antecedent which we have already discussed above. It is not completely clear from [9] what the LJ inference rules for negation should be, but the most reasonable interpretation corresponds to our derivation schemata.

We need to show that the parameters of the metalogic can be used to represent parameters in the proof-theoretic sense – or "eigenvariables". The purpose of an eigenvariable is to represent a completely undetermined term. If we have a (sub-) derivation in LJ, say, where the endsequent contains eigenvariables we can always replace that eigenvariable with any other term without invalidating the derivation. Since all inference rules of the metalogic are closed under substitution of parameters, we can use parameters to represent eigenvariables.

By adding to FOL the clause $T{=}T \Leftarrow$, we can also handle equality. Using this clause, we obtain the following rule for equality in the succedent, for all terms t:

$$\frac{}{\Gamma \vdash t{=}t}\ \vdash D$$

The antecedent rule for equality is more complicated. We can not simply instantiate the $D\vdash$ rule with this clause, since the form of the instance depends on whether the two terms in the equality are unifiable or not. Roughly speaking, we get one alternative for each case. For the case when s and t are not unifiable, we get

$$\frac{}{s{=}t\,,\,\Gamma \vdash C}\ D\vdash$$

In other words, the clause for equality enforces free equality. Equality can never hold between two different parameter-free terms. Since parameters do not represent any particular terms, two different terms with parameters could be equal, i.e. if they are unifiable. If the two terms can not be equal, the assumption of the sequent is absurd and the sequent holds immediately. For the case when $\sigma^*{=}mgu(s,t)$, we get

$$\frac{\Gamma\sigma^* \vdash C\sigma^*}{s{=}t\,,\,\Gamma \vdash C}\ D\vdash$$

By including a specialisation step, we can even obtain:

$$\dfrac{\dfrac{\Gamma(t) \vdash C(t)}{\Gamma(t)\sigma^* \vdash C(t)\sigma^*}\ \text{specialisation}}{s{=}t\,,\,\Gamma(s) \vdash C(s)}\ D\vdash$$

(This derivation schema was the result of a discussion with Peter Schroeder-Heister.)

From these examples, it can be seen that the antecedent rule for $T{=}T \Leftarrow$ becomes a very general substitution principle. Since this clause really expresses a semantic fact about equality, we can not give a fully satisfactory treatment of it using the deductive approach. In section 5, we will use the semantic approach to give a precise meaning to the equality clause.

There is a problem with the system obtained using FOL. The reflection principle applies to all judgments, even those that have no definitional clause. If there is no clause defining a judgment A, the $D\vdash$ rule instance becomes

$$\frac{}{\Gamma, A \vdash C}\ D\vdash$$

e.g. all sequents assuming A will hold immediately, since assuming A would be absurd. To avoid this effect on atomic formulae, we include the clause $g(P) \Leftarrow g(P)$. (Recall that g was the atomic formula tag.) With this clause, the $\vdash D$ and $D\vdash$ rule instances for atomic formulae (except equalities) becomes

$$\frac{\Gamma \vdash A}{\Gamma \vdash A}\ \vdash D \qquad\qquad \frac{\Gamma, A \vdash C}{\Gamma, A \vdash C}\ D\vdash$$

which can not serve any useful purpose in a derivation, since for both the premise and conclusion are identical. In this manner, atomic formulae are made "undefined".

4 Deductive Approach Example: Naive Set Theory

As another example, we modify the definition FOL for reasoning in naive set theory. Our formalisation is based on that of Hallnäs [10], and the example in [12]. To represent formulae of set theory, we extend first-order logic with a logical constant for set membership. The characteristic of naive set theory as opposed to axiomatic set theories is that sets can be formed by comprehension in an unrestricted way, i.e. every predicate on sets $P(x)$, defines a set comprising exactly those elements for which $P(x)$ is true. Thus we can represent any set by the expression $\{x \mid P(x)\}$. We will represent this expression as $setexpr(\lambda x.P(x))$, for some arbitrary unique constant *setexpr*, but will continue to use the expression itself as syntactic sugar.

This representation suggests the following inference rules for set membership

$$\frac{\Gamma \vdash P(a)}{\Gamma \vdash a \in \{x \mid P(x)\}} \vdash D \qquad \frac{\Gamma, P(a) \vdash C}{\Gamma, a \in \{x \mid P(x)\} \vdash C} D \vdash$$

The succedent rule is given by the clause

$$a \in \{x \mid P(x)\} \Leftarrow P(a)$$

and the antecedent rule is obtained from this clause by definitional reflection.

Equality is given by the principle of extensionality, i.e. two sets are equal if they have the same elements. The inference rule for membership in the succedent would be

$$\frac{\Gamma, X^* \in A \vdash X^* \in B \qquad \Gamma, X^* \in A \vdash X^* \in B}{\Gamma \vdash A = B} \vdash D$$

it could be defined by the clause:

$$A = B \Leftarrow \Pi x\ (x \in A \Rightarrow x \in B, x \in B \Rightarrow x \in A)$$

Using reflection, the equality clause can be used to substitute the set A for B in $a \in B$ on either side of the turnstile. For substitution in the succedent, it can be done as:

$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{}{\Gamma, a \in A \vdash a \in A}\ \text{Axiom} \qquad \Gamma \vdash a \in B}{\Gamma, a \in B \Rightarrow a \in A \vdash a \in A}\ \Rightarrow \vdash}{\Gamma, a \in A \Rightarrow a \in B,\ a \in B \Rightarrow a \in A \vdash a \in A}\ \text{Weakening}}{\Gamma, \Pi x (x \in A \Rightarrow x \in B, x \in B \Rightarrow x \in A) \vdash a \in A}\ \Pi \vdash}{\Gamma, A = B \vdash a \in A}\ D \vdash$$

However, the system we have defined so far is not strong enough to substitute for a in $a \in A$. Intuitively, the reason is that $a \in A$ is defined as $P(a)$, for some P. Since a and b could be different objects, even if they are extensionally equal, we can not expect $P(b)$ to follow from $a = b$ and $P(a)$. The obvious solution of adding an axiom such as $\forall A \forall x \forall y\ (x = y \rightarrow (x \in A \leftrightarrow y \in A))$ is not possible since we have to express everything as explicit definitions, and this axiom does not define anything explicitly.

Our solution is to add a new judgment $set(A)$, meaning that A is a "proper" set, i.e that it respects extensional equality among its elements. $set(A)$ would be defined if the statement

of the axiom held for a particular A. The definition of the quantifiers must also be changed to ensure that quantification is only over "proper" sets. The new and changed clauses become

$$set(A) \Leftarrow \Pi x \Pi y\,((set(x), set(y), x{=}y) \Rightarrow (x{\in}A \Rightarrow y{\in}A, y{\in}A \Rightarrow x{\in}A))$$
$$\exists x\,A(x) \Leftarrow A(X), set(X)$$
$$\forall x\,A(x) \Leftarrow \Pi x\,(set(x) \Rightarrow A(x))$$
$$A{=}B \Leftarrow \Pi x\,((set(x), x{\in}A) \Rightarrow x{\in}B, (set(x), x{\in}B) \Rightarrow x{\in}A)$$

Now it is possible to substitute a for b in $b{\in}A$, provided that $set(A)$ holds, as in the derivation fragment:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma, a{\in}A \vdash a{\in}A}\,\text{Axiom} \quad \Gamma \vdash b{\in}A}{\Gamma, b{\in}A \Rightarrow a{\in}A \vdash a{\in}A}\,\Rightarrow\vdash}{\Gamma, a{=}b, b{\in}A \Rightarrow a{\in}A \vdash a{\in}A}\,\text{Weakening}}{\Gamma, a{=}b, a{\in}A \Rightarrow b{\in}A, b{\in}A \Rightarrow a{\in}A \vdash a{\in}A}\,\text{Weakening} \quad \dfrac{}{\Gamma, a{=}b \vdash a{=}b}\,\text{Axiom}}{\Gamma, a{=}b, a{=}b \Rightarrow (a{\in}A \Rightarrow b{\in}A, b{\in}A \Rightarrow a{\in}A) \vdash a{\in}A}\,\Rightarrow\vdash}{\Gamma, a{=}b, \Pi y(a{=}y \Rightarrow (a{\in}A \Rightarrow y{\in}A, y{\in}A \Rightarrow a{\in}A)) \vdash a{\in}A}\,\Pi\vdash}{\Gamma, a{=}b, \Pi x \Pi y(x{=}y \Rightarrow (x{\in}A \Rightarrow y{\in}A, y{\in}A \Rightarrow x{\in}A)) \vdash a{\in}A}\,\Pi\vdash}{\Gamma, a{=}b, set(A) \vdash a{\in}A}\,D\vdash$$

This formulation of naive set theory has the interesting property that although the set-theoretical paradoxes are derivable, the system is not inconsistent. Using e.g. Russell's paradox, it is possible to derive both $\vdash p$ and $\vdash \neg p$. A derivation of this paradox is

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\bot, \{x|\neg x{\in}x\}{\in}\{x|\neg x{\in}x\} \vdash \bot}\,\text{Axiom} \quad \dfrac{}{\{x|\neg x{\in}x\}{\in}\{x|\neg x{\in}x\} \vdash \{x|\neg x{\in}x\}{\in}\{x|\neg x{\in}x\}}\,\text{Axiom}}{\{x|\neg x{\in}x\}{\in}\{x|\neg x{\in}x\} \Rightarrow \bot, \{x|\neg x{\in}x\}{\in}\{x|\neg x{\in}x\} \vdash \bot}\,\Rightarrow\vdash}{\neg\{x|\neg x{\in}x\}{\in}\{x|\neg x{\in}x\}, \{x|\neg x{\in}x\}{\in}\{x|\neg x{\in}x\} \vdash \bot}\,D\vdash}{\{x|\neg x{\in}x\}{\in}\{x|\neg x{\in}x\}, \{x|\neg x{\in}x\}{\in}\{x|\neg x{\in}x\} \vdash \bot}\,D\vdash}{\{x|\neg x{\in}x\}{\in}\{x|\neg x{\in}x\} \vdash \bot}\,\text{Contraction}}{\vdash \{x|\neg x{\in}x\}{\in}\{x|\neg x{\in}x\} \Rightarrow \bot}\,\vdash\Rightarrow}{\vdash \neg\{x|\neg x{\in}x\}{\in}\{x|\neg x{\in}x\}}\,\vdash D}{\vdash \{x|\neg x{\in}x\}{\in}\{x|\neg x{\in}x\}}\,\vdash D$$

Let $\mathcal{P}$ be $\{x \mid \neg x{\in}x\}{\in}\{x \mid \neg x{\in}x\}$. The endsequent of this derivation is $\vdash \mathcal{P}$ and the next-to-last sequent is $\vdash \neg\mathcal{P}$.

Since we do not have the cut rule this does not imply that every sequent of the form $\vdash A$ is derivable. In fact in Hallnäs' natural deduction formalisation of naive set theory [10], there are no normal derivations of $\bot$. Since normal derivations in natural deduction correspond to cut-free derivations in sequent calculus, there can not be any derivation of $\vdash \bot$ in our representation of naive set theory. In other words, there is a "localisation" of the paradoxes that makes it safe to demonstrate set-theoretic results using our representation of naive set theory. In a more general setting [12], Hallnäs shows that the cut rule is admissible in the theory of partial inductive definitions, precisely in those cases where the cut formula is non-paradoxical, i.e. has well-defined (or no) truth value.

5 The Semantic Approach

In contrast to the deductive approach, where the emphasis was on inference rules, the semantic approach puts the emphasis on the semantics of the logic. The idea is to represent a given semantics (notion of truth) using a partial inductive definition. In this section, we will discuss truth of judgments under a particular interpretation. Logical truth will be treated in the next section.

To begin with, we will illustrate the basic ideas of the semantic approach, using classical first-order logic as an example. In that logic, a formula is expressed in a formal first-order language. That formula is given meaning by being mapped into some mathematical structure. Such an *interpretation* maps the predicates, functions and constants of the formal language to predicates, functions and individual objects in a mathematical sense. The interpretation is used to compute a truth value for any atomic formula, by mapping that formula to a mathematical predicate which will be either true or false. In addition, the interpretation defines the set of individuals over which variables can be quantified.

Together with a truth function that defines the meaning of the logical connectives (i.e. truth or falsity of non-atomic formulae), such an interpretation permits a determination of the truth or falsity of arbitrary formulae. It is common in logic to include the truth function in the interpretation, but as the truth function is always the same for a particular logic, we have made it a separate entity.

Since we are representing logics within a completely formal system, we need a way of determining the truth value of a formula without any references to the mathematical structure. Instead of determining truth values of atomic formulae using the mapping, we could do it using a set of all true atomic formulae. Defining that set would require a reference to the mapping, but once its members are known, no further reference to the mapping would be required. An atomic formula would then be true if and only if it is included in that set. For convenience, in the sequel we will call such a set the *interpretation*, although strictly speaking an interpretation is the mapping of formulae into the mathematical structure.

Of course, when representing a logic, one does not have to define a proper interpretation first and use it to derive the set of true judgments and the truth function – that set could be defined directly from an understanding of on how the particular logic works.

The truth functions would also need to be reexpressed using this way of defining truth values. In particular, quantification can no longer be done over the set of individuals, but must be done over the set of formal terms of first-order logic. Since all interpretations define the logical connectives in the same way, the reexpressed truth function will be the same for any interpretation.

To ensure that quantification over terms does not omit any individual, every individual must be the image under the mapping of some term. If some individuals are not, we add new expressions to the formula language to represent each such individual. In the sequel, we will always assume that the interpretations are written so that this requirement is fulfilled. This is no real restriction, as we could assume that the formula language had a special set of terms set aside for that purpose and that the interpretation was extended, as necessary, to map these terms onto the individuals that would not otherwise be accessible (or to an arbitrary individual if all individuals were already accessible).

Since we use a countably infinite language to express terms, this way of eliminating the set of individuals is possible only if that set is also countable, so the semantic approach is limited to representing countable interpretations. However, when we later represent the notion of "logical truth" (truth under all interpretations), truth in uncountable interpretations will be included, since the interpretations need not actually be constructed.

In general, several terms may correspond to one individual, e.g. the interpretation may map both the terms 1+1 and 2 to the single numeral "2". This does not pose any problems as for any true formula containing the term 2, there must be another one containing the term 1+1 at the same position, and vice versa. It does not matter which one of the terms is used, so quantification can be done uniformly over the set of terms – regardless of the set of individuals of the interpretation.

The concept of interpretation is used in a similar way also for other logics than classical first-order logic. However, we will generalise the concept of a formula, talking about general judgments instead.

Judgments are divided into atomic and non-atomic ones. Atomic judgments are those where the truth values are determined by the interpretation, while the truth values of non-atomic judgments are determined by the truth values of their constituent atomic judgments according to the truth function. The difference between atomic judgments and atomic conditions should be noted – the latter are atomic units of reasoning of the metalogic. Just as with atomic formulae, we will use a tagging scheme in the representation of atomic judgments.

For logics other than classical first-order logic we will eliminate references to the interpretation in a similar manner as above, defining a set of true atomic judgments and a number of sets of terms, one for each set of individuals.

To represent truth in a particular interpretation, we will use a partial inductive definition consisting of two parts, the interpretation part (I) and the semantic part (T). When there is no risk of misunderstanding, we will simply refer to I as the "interpretation" and T as the "semantics". The interpretation part defines the true atomic judgments.

To be able to distinguish between different judgment forms easily, we use the convention that all judgments are written in the form $j(e_1,\ldots,e_n)$, where j determines the judgment form. For brevity, we will generally write judgments as $j(e)$, with a single argument. It should always be obvious how to generalise this to judgments with more than one argument (or even without arguments). As we mentioned before, when there is only one judgment form, j, it could be convenient to identify the judgment $j(e)$ with e, effectively dropping j and and simply using e itself as the judgment.

If j is an atomic judgment, we have $\vdash_I j$ if and only if j is true. Since no atomic judgment can be both true and false under a given interpretation, we require that any I that represents the actual interpretation has the property that not both $\vdash_I j$ and $j \vdash_I \bot$. This requirement is fulfilled if I is total. If I is also complete, this means that for any atomic judgment j, we have $j \vdash_I \bot$ if and only if j is false according to I. Although I should intuitively be a complete definition, it is possible to have a non-complete I. Completeness of the representation may then suffer – i.e. there may be true (or false) judgments that can not be shown to be true (or false) – but soundness will be preserved, i.e. no judgments can incorrectly be shown to be true or false. This can be seen in the conditions of the correctness theorem below.

We will require that the properties of the previous paragraph hold even when the semantic part of the partial inductive definition is added, i.e. when the derivability relation is $\vdash_{I \cup T}$ rather than $\vdash_I$. A simple way to ensure this is to write T so that it does not include any clause defining an atomic judgment, and to write I so that no clause of it contains a condition that is a non-atomic judgment, either in the head or in the body. These conditions also apply to occurrences of terms that may be instantiated to atomic or non-atomic judgments, respectively.

Since we represent interpretations using a formal notation – clauses of partial inductive definitions – it is a fundamental consequence that we can only represent interpretations where the set of true atomic judgments is recursively enumerable (decidable or semidecidable). If this set is semidecidable, it will be unavoidable that I is not complete (otherwise $\vdash_I$ would decide I).

The set of terms will be implicitly represented by determining the type of the variables in T that range over terms.

In the simplest case an interpretation could be a set of definitional clauses enumerating the true atomic judgments, e.g.

$$p \Leftarrow$$
$$q \Leftarrow$$
$$X{=}X \Leftarrow$$

defines an interpretation where the true judgments are p, q, and all equalities between identical terms, all other judgments are false. This partial inductive definition is both complete and total. The interpretation could just as well be a more complicated definition, in which case the writer of the definition must ensure that it has the desired properties.

Since nothing in the soundness of the semantic approach (which is discussed below) depends on the conditions on I we have just given, it would be possible to have an I where $\vdash_I j$ and $j \vdash_I \bot$ could both hold – corresponding to an "interpretation" where j was true and false at the same time. Conversely, if I is not complete, it can be the case that neither $\vdash_I j$ nor $j \vdash_I \bot$ holds. According to our discussion above, this should be understood as a case where j is false (since not $\vdash_I j$), but where that could not be shown to hold. A different view could be that j is neither true nor false, but lacks a truth value. In both these cases, I does not represent an actual interpretation. We will call such an I a *pseudo-interpretation*. It turns out that pseudo-interpretations can be both meaningful and useful. In the next section we will see examples of the use of pseudo-interpretations.

The semantic part T of the partial inductive definition represents the truth values of non-atomic judgments by representing the truth functions of judgments given by the semantics of the object logic. To obtain the truth function definitions from T, first partition T into subdefinitions T_j for each judgment form j – i.e. exactly those clauses defining a particular judgment form should all be in the same subdefinition. If there is only one judgment form, there will only be one subdefinition, identical to T. Each subdefinition is interpreted as corresponding to a truth function $\mathcal{T}_j$ for the particular judgment form, such that

$$\mathcal{T}_j(X) = \begin{cases} \mathcal{T}[T_j] & \text{If } j(X) \text{ is a non-atomic judgment} \\ \mathit{true} & \text{If } j(X) \text{ is an atomic judgment that is true according to the interpretation.} \\ \mathit{false} & \text{If } j(X) \text{ is an atomic judgment that is false according to the interpretation.} \end{cases}$$

Here $\mathcal{T}[]$ is a syntactic mapping given by

$\mathcal{T}[\{\}]$	*false*	
$\mathcal{T}[\{K_1,K_2,\ldots,K_n\}]$	$\mathcal{T}[K_1]$ *or* ... *or* $\mathcal{T}[K_n]$	(where $n>0$)
$\mathcal{T}[j(H)\Leftarrow B]$	*some* $x_1,\ldots,x_n$ (*equal*(X,H) *and* $\mathcal{T}[B]$) (where $x_1,\ldots,x_n$ are the free variables in $j(H)\Leftarrow B$)	
$\mathcal{T}[(\,)]$	*true*	
$\mathcal{T}[(C_1,C_2,\ldots,C_n)]$	$\mathcal{T}[C_1]$ *and* ... *and* $\mathcal{T}[C_n]$	(where $n>0$)
$\mathcal{T}[C\Rightarrow C']$	*not* $\mathcal{T}[C]$ *or* $\mathcal{T}[C']$	
$\mathcal{T}[\Pi x\, C(x)]$	*every* x: $\mathcal{T}[C(x)]$	
$\mathcal{T}[\perp]$	*false*	
$\mathcal{T}[j'(a)]$	$\mathcal{T}_{j'}(a)$ (where $j'(a)$ is a judgment, i.e. an atomic *condition*)	

Note that since $\mathcal{T}[]$ defines a syntactic transformation, the variable X in the mapping of $\mathcal{T}[H\Leftarrow B]$ is the same one as in the definition of $\mathcal{T}_j$, i.e. the argument variable of $\mathcal{T}_j$.

true and *false* are truth values. The primitive truth functions *and*, *or*, *not*, *every*, and *some*, are defined as follows:

x *and* y	*true* if both x and y are *true*, *false* otherwise.
x *or* y	*true* if either x or y is *true*, *false* otherwise.
not x	*true* if x is *false*, *false* otherwise.
every x:$P(x)$	*true* if $P(a)$ is *true* for every a of the same type as x, *false* otherwise.
some x:$P(x)$	*true* if $P(a)$ is *true* for some a of the same type as x, *false* otherwise.
$X=Y$	*true* if X is equal to Y, *false* otherwise.

For every definition T, we will also define a general truth function for all judgment forms, $\mathcal{T}$, using the equation $\mathcal{T}(j(e)) = \mathcal{T}_j(e)$.

We will apply this transformation to the definition FOL, in order to show that it represents the semantics of first-order predicate logic. The judgment $t(A)$ will denote that A is a true formula under some interpretation. We apply the convention that the judgment $t(A)$ is identified with the formula A. The truth function corresponding to FOL is:

$\mathcal{T}_t(X)$ = *some* A,B: (*equal*$(X,A\wedge B)$ *and* $\mathcal{T}_t(A)$ *and* $\mathcal{T}_t(B)$)
or some A,B: (*equal*$(X,A\vee B)$ *and* $\mathcal{T}_t(A)$)
or some A,B: (*equal*$(X,A\vee B)$ *and* $\mathcal{T}_t(B)$)
or some A,B: (*equal*$(X,A\rightarrow B)$ *and not* $\mathcal{T}_t(A)$ *or* $\mathcal{T}_t(B)$)
or some A: (*equal*$(X,\neg A)$ *and not* $\mathcal{T}_t(A)$ or *false*)
or some A,x: (*equal*$(X,\exists x\, A(x))$ *and* $\mathcal{T}_t(A(x))$)
or some A: (*equal*$(X,\forall x\, A(x))$ *and every* x: $\mathcal{T}_t(A(x))$)

$\mathcal{T}_t(X)$ = *true* If X is an atomic formula that is true according to the interpretation.

$\mathcal{T}_t(X)$ = *false* If X is an atomic formula that is false according to the interpretation

By using the obvious algebraic properties of the primitive truth functions to do some natural simplifications, the function can be written more clearly. In the sequel, we will do such simplifications without comment.

$\mathcal{T}_t(A\wedge B) = \mathcal{T}_t(A)$ *and* $\mathcal{T}_t(B)$
$\mathcal{T}_t(A\vee B) = \mathcal{T}_t(A)$ *or* $\mathcal{T}_t(B)$
$\mathcal{T}_t(A\rightarrow B) =$ *not* $\mathcal{T}_t(A)$ or $\mathcal{T}_t(B)$
$\mathcal{T}_t(\neg A) =$ *not* $\mathcal{T}_t(A)$
$\mathcal{T}_t(\exists x\, A(x)) =$ *some* x $\mathcal{T}_t(A(x))$

$\mathcal{T}_t(\forall x\, A(x)) = every\ x\ \mathcal{T}_t(A(x))$
$\mathcal{T}_t(A) = true$ if A is an atomic formula, true according to the interpretation
$\mathcal{T}_t(A) = false$ otherwise

It should be clear that this function faithfully expresses the truth of a formula in first-order predicate logic.

Of course, not every partial inductive definition corresponds to a meaningful truth function. For example, letting $j(a)$ be a non-atomic judgment, a definition containing the single clause $j(a) \Leftarrow j(a) \Rightarrow \bot$, corresponds to the (slightly simplified) function

$$\mathcal{T}_j(X) = \begin{cases} equal(X, a)\ and\ not\ \mathcal{T}_j(X) & \text{If } j(X) \text{ is a non-atomic judgment} \\ true & \text{If } j(X) \text{ is an atomic judgment that is true according to the interpretation.} \\ false & \text{If } j(X) \text{ is an atomic judgment that is false according to the interpretation.} \end{cases}$$

which is not well-defined for $\mathcal{T}_j(a)$. Since the representation of an object logic using the semantic approach involves constructing a partial inductive definition that corresponds to a *given* truth function of the logic in question, the possibility of meaningless truth functions need not concern us. We must presume that the object logic was initially formulated with a meaningful semantics. On the other hand, to obtain a soundness result for the mapping of partial inductive definitions to functions, we must define precisely what we mean by a "meaningful" truth function.

The requirement we need is that the function is *total*, i.e. for every judgment J, we must have either $\mathcal{T}(J)=true$ or $\mathcal{T}(J)=false$. Actually, since we have assumed that the truth values of atomic judgments are well-defined by the interpretation, all we need is the weaker requirement that $\mathcal{T}(J)$ is total under the assumption that it is total for atomic judgments. This weaker requirement is easier to verify, since it depends only on that part of the truth function that expresses the truth of non-atomic judgments – the part corresponding to T.

We interpret a sequent $\Gamma \vdash_{I \cup T} C$, where the succedent and the elements of the antecedent are all judgments, as follows: According to the truth function corresponding to T with atomic judgments having truth values according to I, either the truth value of C is *true* (i.e. $\mathcal{T}(C)=true$) or the truth value of one of the judgments in Γ is *false*. We have now arrived at our goal of being able to represent a logic and an interpretation, and being able to express that judgments are true or false in that logic according to the interpretation. The following theorem shows that the representation correctly expresses what we intend.

THEOREM Correctness of the representation
Let J be a judgment, Γ be a set of judgments, I be a partial inductive definition representing an interpretation and T be a partial inductive definition representing a semantics, such that $\mathcal{T}$ is a total function from judgments to truth values. Then

(soundness) If $\Gamma \vdash_{I \cup T} J$ holds, then $\mathcal{T}(J)=true$, or – for some $G \in \Gamma$ – $\mathcal{T}(G)=false$.

If, in addition, $I \cup T$ is a complete partial inductive definition, then also

(completeness) If $\mathcal{T}(J)=true$, or – for some $G \in \Gamma$ – $\mathcal{T}(G)=false$, then $\Gamma \vdash_{I \cup T} J$ holds.

The proof is omitted for reasons of space. The interested reader is referred to [6]. ∎

An immediate corollary is that when $I \cup T$ is complete, $\vdash_{I \cup T} J$ holds iff $\mathcal{T}(J)=true$, and $J \vdash_{I \cup T} \bot$ holds iff $\mathcal{T}(J)=false$.

Given the partial inductive definition FOL and an interpretation where every atomic formulae is false, i.e. an empty interpretation, we have $\vdash \neg a$, $\vdash b \rightarrow b$, $\vdash a \rightarrow b$, and $\vdash (a \wedge b) \rightarrow \neg (\neg a \vee \neg b)$ but $\nvdash a$ and $\nvdash \exists x\, r(x)$.

If we wanted an interpretation where the atomic formulae a and $r(1)$ are true, but no other, we can add the two clauses $a \Leftarrow$ and $r(1) \Leftarrow$. We then have $\vdash a$, $\vdash \exists x\, r(x)$, $\vdash b \rightarrow b$, and $\vdash (a \wedge b) \rightarrow \neg(\neg a \vee \neg b)$ but $\nvdash \neg a$ and $\nvdash a \rightarrow b$.

To obtain first-order logic with (free) equality, the interpretation must include all formulae of the form $t=t$, and no formulae of the form $s=t$, where s and t are different terms. A partial inductive definition representing such an interpretation could define the equality predicate with the single clause:

$$T=T \Leftarrow$$

6 Logical Truth

Often the question of logical truth, i.e. truth under all interpretations, is at least as much of interest as the question of truth under a given interpretation. The semantic approach can be used also for showing the logical truth of judgments.

When we are talking about an "arbitrary interpretation", two different things are involved. One is the arbitrary mapping of atomic judgments to mathematical predicates, the other is the arbitrary domain of quantified variables. From the discussion in the previous section, we recall that any combination of these two things defines some set of true atomic judgments – the domain of quantification in the truth function in all cases being the set of terms. This implies that to show truth under all interpretations, it suffices to show truth under all sets of true atomic judgments – or "interpretations" in our terminology.

Since interpretations are represented by partial inductive definitions, we can not quantify over them. Instead, we use a pseudo-interpretation consisting of only the clauses

$$j(g(X)) \Leftarrow j(g(X))$$

for all judgment forms j, where g is the appropriate "tag" used for atomic judgments. We call such a definition L. L is a pseudo-interpretation where no judgments have a truth value, i.e. neither $\vdash_L j(a)$ nor $j(a) \vdash_L \bot$ holds for any $j(a)$. Intuitively, we would expect (non-atomic) judgments that are true in this pseudo-interpretation, to be logically true, since their truth value can not depend on the truth or falsity of any particular atomic judgment and thus not on any particular interpretation.

We can show formally that this is the case. Consider a $\vdash$D or D$\vdash$ inference step where the condition a of the presentation of these inference rules represents an atomic judgment. Since such an inference step would use one of the clauses $j(g(X)) \Leftarrow j(g(X))$, its premise would be identical to the conclusion, i.e. the inference step would be completely redundant and could just as well be removed from the derivation. In other words, a derivation of a judgment using the definition $T \cup L$ does not depend in any way on the actual contents of L. L could just as well be replaced by an arbitrary ordinary interpretation – that is, the judgment is true in all interpretations.

Note that even though we are unable to represent interpretations that are not recursively enumerable, L still gives us truth in all interpretations – including those where the set of all true judgments is not recursively enumerable, or even countable – since no arbitrary interpretation needs to be actually represented as a definition.

With the partial inductive definition FOL and the pseudo-interpretation L, we have $\vdash (a \wedge b) \rightarrow \neg(\neg a \vee \neg b)$ but $\nvdash a$ and $\nvdash \neg a$.

We can also write pseudo-interpretations where some atomic formulae have defined truth values, while others do not. To do this the clause $j(g(X)) \Leftarrow j(g(X))$ should be changed to cover only those atomic formulae which should not have defined truth values.

Unfortunately, it is not always the case that $\vdash_{T \cup L} J$ holds if the judgment J is logically true. This is only to be expected since $T \cup L$ is not a complete definition, so the completeness part of the correctness theorem is not applicable.

Consider again the representation of the semantics of first-order logic and the judgment $p \vee \neg p$. Even though this judgment clearly holds under any interpretation, it does not hold in the pseudo-interpretation L. The reason is that even though this judgment is true in all interpretations, the derivation of it requires an actual truth value for p, either true or false. That is, different derivations are required in different interpretations. Depending on whether p is true or false in a particular interpretation I, we will have one of the two following derivation fragments:

$$\frac{\vdash p}{\vdash p \vee \neg p}\ \vdash \mathrm{D} \qquad\qquad \frac{\dfrac{\dfrac{p \vdash \bot}{\vdash p \Rightarrow \bot}\ \vdash \Rightarrow}{\vdash \neg p}\ \vdash \mathrm{D}}{\vdash p \vee \neg p}\ \vdash \mathrm{D}$$

Every time there is a choice of how to compute the truth value of a judgment, the possibility of different derivations arise. In the definition of first-order logic, the only place where such a choice can be made is in the definition of the $\vee$ connective. Those logically true judgments we can derive are those where the choice can be made uniformly, independent of the interpretation.

Consider also the judgment $\neg \forall x\ p(x) \rightarrow \exists x\ \neg p(x)$. This judgment is true in any interpretation, still it does not hold in the pseudo-interpretation L. As in the case with $p \vee \neg p$, the problem is that a choice must be made. This time the choice is of an actual instance of the existentially quantified x. Such as choice can not be made, since it does not follow from $\neg \forall x\, p(x)$ that $p(a)$ is false for any particular a.

This is closely related to the ideas of intuitionistic logic, where all derivations must be constructive, i.e. the derivation of a disjunctive formula must be justified with the derivation of *one* of the parts of the disjunction. The formula $p \vee \neg p$ is not derivable in intuitionistic logic, since neither p nor $\neg p$ is derivable. Similarly, the derivation of an existentially quantified formula is justified with the derivation of an actual instance of that formula. Thus we could be justified in saying that, in a sense, with the pseudo-interpretation L we obtain truth in an "intuitionistic" version of the logic defined by T.

Indeed, in the case of first-order logic we do get exactly the intuitionistic notion of truth. Using the deductive approach we can see that the partial inductive definition FOL gives the inference rules for Gentzen's intuitionistic sequent calculus LJ.

It is important that there are as few alternative definitional clauses as possible. In FOL, implication is defined by the clause $A \to B \Leftarrow A \Rightarrow B$. It would have been just as correct to give the two clauses $A \to B \Leftarrow A \Rightarrow \bot$ and $A \to B \Leftarrow B$. The truth functions given by $\mathcal{T}[]$ in the two cases would have been equivalent. However, this would introduce another choice between two alternatives – further reducing the set of logically true judgments that could be derived. In that case we would not even have the advantage of an established logic to classify those judgments that are still derivable.

It is possible to recover completeness by adding to L the following clauses:

$$\mathit{classical} \Leftarrow \Pi x\, \mathit{classical}'(x)$$
$$\mathit{classical}'(X) \Leftarrow X$$
$$\mathit{classical}'(X) \Leftarrow X \Rightarrow \bot$$

and including *classical* in the antecedent of the sequent when a judgment should be derived. *classical* is defined by these clauses to hold if every atomic judgment is either true or false. Assuming *classical* implies assuming that every atomic judgment has a well-defined truth value. With *classical*, we can get derivation fragments such as this:

$$\dfrac{\dfrac{\dfrac{p \vdash J \qquad p \Rightarrow \bot \vdash J}{\mathit{classical}'(p) \vdash J}\ \mathrm{D}\vdash}{\Pi x\, \mathit{classical}'(x) \vdash J}\ \Pi\vdash}{\mathit{classical} \vdash J}\ \mathrm{D}\vdash$$

To complete this derivation, two sequents need to be derived, each with the judgment J as succedent. One sequent has the assumption that the judgment p is true, the other has the assumption that the judgment p is false. In other words, the derivation has been divided into two cases where p has the truth values true and false, respectively. Using contraction on *classical* the derivation of A can be divided into truth/falsity cases for an arbitrary number of atomic judgments. In this way all logically true judgments can be derived. We can introduce a new judgment form *ctrue*, for classical truth, defined by the clause

$$\mathit{ctrue}(A) \Leftarrow \mathit{classical} \Rightarrow A$$

Letting LC be the partial inductive definition obtained by taking L together with the clauses for *classical*, *classical'* and *ctrue*, we have $\vdash_{T \cup LC} \mathit{ctrue}(A)$ iff A is logically true.

$\mathit{ctrue}(p \vee \neg p)$ can be shown to hold by the following derivation using the definition $T \cup LC$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\bot, p \vdash \bot}\ \text{Axiom} \qquad \dfrac{}{p \vdash p}\ \text{Axiom}}{p \Rightarrow \bot, p \vdash \bot}\ \Rightarrow\vdash}{p \Rightarrow \bot \vdash p \Rightarrow \bot}\ \vdash\Rightarrow}{p \Rightarrow \bot \vdash \neg p}\ \vdash\mathrm{D}}{p \Rightarrow \bot \vdash p \vee \neg p}\ \vdash\mathrm{D} \qquad \dfrac{p \vdash p}{p \vdash p \vee \neg p}\ \vdash\mathrm{D}}{\mathit{classical}'(p) \vdash p \vee \neg p}\ \mathrm{D}\vdash}{\Pi x\, \mathit{classical}'(x) \vdash p \vee \neg p}\ \Pi\vdash}{\mathit{classical} \vdash p \vee \neg p}\ \mathrm{D}\vdash}{\vdash \mathit{classical} \Rightarrow p \vee \neg p}\ \vdash\Rightarrow}{\vdash \mathit{ctrue}(p \vee \neg p)}\ \vdash\mathrm{D}$$

Although more complicated (*classical* must be used twice), a derivation of $ctrue(\neg\forall x\, p(x) \rightarrow \exists x \neg p(x))$ can be done in essentially the same way ([6] example 2.3.1).

7 Examples of the Semantic Approach

We will illustrate the semantic approach by two examples. The logics we will represent in this section are modal logics and three-valued logics.

We first carry out the representation of modal logic using Kripke semantics. In Kripke semantics, an interpretation is typically given as a triple $\langle W,R,V\rangle$, where W is a set of possible worlds, R is an accessibility relation between worlds and V is a binary relation between worlds and formulae, defining which formulae are true in each world. In other words, the truth of a formula is dependent on the world in which this formula is considered. The modal operators $\Box$ (necessity) and $\Diamond$ (possibility) are defined in terms of how it is possible to move from the current world to other worlds, as defined by the relation R. A formula is necessarily true in some world, if the formula inside the modality is true in all directly accessible worlds. A formula is possibly true in some world, if the formula inside the modality is true in at least one of the accessible worlds. A formula is simply true, if it is true in all the worlds in W.

We will represent formulae of the modal logic in the same way as formulae of the propositional part of first-order logic. Since nothing in Kripke semantics depends on what the worlds themselves actually are, we leave the representation of worlds unspecified. We get three basic judgment forms:

- $W::A$ The relation V, i.e. the formula A is true in the world W.
- $W\ll W'$ The relation R, i.e. the world W' is accessible from the world W.
- $true(A)$ The formula A is true (in every world).

The judgment form $W\ll W'$ has only atomic judgments, i.e. it is not defined by the semantic definition. The judgment form $true(A)$ has no atomic judgments. Judgments of the form $W::A$ are atomic iff A is a proposition letter. The interpretation (in our sense of the word) would define completely the judgment form $W\ll W'$ and the judgments $W::A$, where A is a proposition letter.

Letting the variables W, W' and w have the type of possible worlds, a definition that gives the semantics of these judgments is

$$W::A\wedge B \Leftarrow W::A,\ W::B$$
$$W::A\vee B \Leftarrow W::A$$
$$W::A\vee B \Leftarrow W::B$$
$$W::A\rightarrow B \Leftarrow W::A \Rightarrow W::B$$
$$W::\neg A \Leftarrow W::A\Rightarrow\bot$$
$$W::\Box A \Leftarrow \Pi w\ (W\ll w \Rightarrow w::A)$$
$$W::\Diamond A \Leftarrow W\ll W',\ W'::A$$
$$true(A) \Leftarrow \Pi w\ w::A$$

Note that here $\rightarrow$ denotes ordinary (logical) implication, not strict implication.

We will show that the truth functions corresponding to this particular definition using the mapping $\mathcal{T}[]$ do give the correct interpretation of the modal connectives. The clauses for $\Box$ and $\Diamond$ define

$\mathcal{T}_{::}(W, \Box X)$ = *every* w: (*not* $\mathcal{T}_{«}(W, w)$ *or* $\mathcal{T}_{::}(w, X)$)
$\mathcal{T}_{::}(W, \Diamond X)$ = *some* W': ($\mathcal{T}_{«}(W, W')$ *and* $\mathcal{T}_{::}(W', X)$)

which is indeed the correct definition according to Kripke semantics.

For logical truth (truth under all interpretation), the situation is different than that for non-modal logics, as there are typically restrictions on the accessibility relation R. By considering only such interpretations where R has certain properties, e.g. reflexivity, logical truth in different modal systems are obtained.

A pseudo-interpretation for logical truth similar to the ones we have seen previously would have the form

$W::g(A) \Leftarrow W::g(A)$
$W«W' \Leftarrow W«W'$

However, such a pseudo-interpretation would just give logical truth in the system K, where there are no conditions on R. To obtain logical truth in the other systems, we will add clauses that force the judgment form $W«W'$ to have the desired properties. E.g. by adding to the pseudo-interpretation the two clauses

$W«W \Leftarrow$
$W«W' \Leftarrow W«W'', W''«W'$

R would be forced to be a reflexive and transitive relation, so truth in the modal system S4 would be obtained.

One of the characteristic axioms of S4, $\Box p \rightarrow \Box\Box p$, can be shown to be logically true in S4 by the following derivation:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{W'''^*::p,\ W^*«W'^*,\ W'^*«W'''^* \vdash W'''^*::p}\ \text{Axiom} \quad \dfrac{}{\begin{matrix}W^*«W'^*,\\ W'^*«W'''^*\\ \vdash W^*«W'^*\end{matrix}}\ \text{Axiom} \quad \dfrac{}{\begin{matrix}W^*«W'^*,\\ W'^*«W'''^*\\ \vdash W'^*«W'''^*\end{matrix}}\ \text{Axiom}}{W^*«W'^*,\ W'^*«W'''^* \vdash W^*«W'''^*}\ \vdash D}{W^*«W'''^* \Rightarrow W'''^*::p,\ W^*«W'^*,\ W'^*«W'''^* \vdash W'''^*::p}\ \Rightarrow\vdash}{\Pi w(W^*«w \Rightarrow w::p),\ W^*«W'^*,\ W'^*«W'''^* \vdash W'''^*::p}\ \Pi\vdash}{W^*::\Box p,\ W^*«W'^*,\ W'^*«W'''^* \vdash W'''^*::p}\ D\vdash}{W^*::\Box p,\ W^*«W'^* \vdash W'^*«W'''^* \Rightarrow W'''^*::p}\ \vdash\Rightarrow}{W^*::\Box p,\ W^*«W'^* \vdash \Pi w(W'^*«w \Rightarrow w::p)}\ \vdash\Pi}{W^*::\Box p,\ W^*«W'^* \vdash W'^*::\Box p}\ \vdash D}{W^*::\Box p \vdash W^*«W'^* \Rightarrow W'^*::\Box p}\ \vdash\Rightarrow}{W^*::\Box p \vdash \Pi w(W^*«w \Rightarrow w::\Box p)}\ \vdash\Pi}{W^*::\Box p \vdash W^*::\Box\Box p}\ \vdash D}{\vdash W^*::\Box p \Rightarrow W^*::\Box\Box p}\ \vdash\Rightarrow}{\vdash W^*::\Box p \rightarrow \Box\Box p}\ \vdash D}{\vdash \Pi w\, w::\Box p \rightarrow \Box\Box p}\ \vdash\Pi}{\vdash true(\Box p \rightarrow \Box\Box p)}\ \vdash D$$

Just as with predicate logic, we have only obtained truth in the intuitionistic sense. Classical truth can be obtained using similar techniques:

$$classical \Leftarrow \Pi w\, \Pi x\, classical'(w, x)$$
$$classical'(W, X) \Leftarrow W::X$$
$$classical'(W, X) \Leftarrow W::X \Rightarrow \bot$$

$$ctrue(A) \Leftarrow classical \Rightarrow true(A)$$

The intention of the next example is to show how the semantic approach can be extended in a natural way to logics with more than two truth values. We will represent the three-valued logics of Kleene and Lukasiewicz [15]. The example was inspired by the representation of three-valued logic in LF described in [1].

Kleene's logic has three truth values: *t*, *f*, and *u*. The first two are the usual truth and falsity values, while *u* stands for a particular truth value meaning "undefined". The meaning of the logical connectives of Kleene's logic is given by the following truth tables. The rows correspond to different first arguments and the columns to different second arguments of the connective in the top-left corner.

$\neg$	
t	f
f	t
u	u

$\vee$	t	f	u
t	t	t	t
f	t	f	u
u	t	u	u

$\wedge$	t	f	u
t	t	f	u
f	f	f	f
u	u	f	u

$\rightarrow$	t	f	u
t	t	f	u
f	t	t	t
u	t	u	u

$\equiv$	t	f	u
t	t	f	u
f	f	t	u
u	u	u	u

The logic of Lukasiewicz has different tables for implication and equivalence. The connectives for that logic will be written $\rightarrow_L$ and $\equiv_L$. Additionally, we introduce an "external" equivalence connective, $\cong$, which holds between two expressions that have equal truth values under all interpretations.

$\rightarrow_L$	t	f	u
t	t	f	u
f	t	t	t
u	t	u	t

$\equiv_L$	t	f	u
t	t	f	u
f	f	t	u
u	u	u	t

$\cong$	t	f	u
t	t	f	f
f	f	t	f
u	f	f	t

The three truth values are represented using two different judgment forms, *true*(*e*) and *false*(*e*). If *e* has truth value *t* or *f*, the appropriate judgment holds. If it has the truth value *u*, none of them hold. The semantics of each judgment is obtained by splitting each truth table in two; one for each judgment form. For the connective $\rightarrow$, we get the following two tables. (We enter *Y* if the judgment should hold and *N* if it should not, to avoid confusion with the truth values).

true $\rightarrow$	t	f	u
t	Y	N	N
f	Y	Y	Y
u	Y	N	N

false $\rightarrow$	t	f	u
t	N	Y	N
f	N	N	N
u	N	N	N

By deciding in this way when each judgment form should hold for every connective, we can construct a partial inductive definition that gives the appropriate truth functions. To improve the readability of that definition, we will identify the judgment form *true*(*e*) with *e*. Also, since according to the truth table for negation, *false*(*e*) holds iff *true*($\neg e$) holds, we can identify the judgment form *false*(*e*) with $\neg e$.

$A \wedge B \Leftarrow A, B$
$A \vee B \Leftarrow A$
$A \vee B \Leftarrow B$
$A \rightarrow B \Leftarrow (\neg A \Rightarrow \bot) \Rightarrow B$
$A \equiv B \Leftarrow (\neg A \Rightarrow \bot) \Rightarrow B, (A \Rightarrow \bot) \Rightarrow \neg B$
$A \rightarrow_L B \Leftarrow A \Rightarrow B, \neg B \Rightarrow \neg A$
$A \equiv_L B \Leftarrow A \Rightarrow B, B \Rightarrow A, \neg A \Rightarrow \neg B, \neg B \Rightarrow \neg A$
$A \equiv B \Leftarrow A \Rightarrow B, B \Rightarrow A, \neg A \Rightarrow \neg B, \neg B \Rightarrow \neg A$

$\neg(A \wedge B) \Leftarrow \neg A,$
$\neg(A \wedge B) \Leftarrow \neg B$
$\neg(A \vee B) \Leftarrow \neg A, \neg B$
$\neg(A \rightarrow B) \Leftarrow A, \neg B$
$\neg(A \equiv B) \Leftarrow (\neg A \Rightarrow \bot) \Rightarrow \neg B, (B \Rightarrow \bot) \Rightarrow A$
$\neg(A \rightarrow_L B) \Leftarrow A, \neg B$
$\neg(A \equiv_L B) \Leftarrow (\neg A \Rightarrow \bot) \Rightarrow \neg B, (B \Rightarrow \bot) \Rightarrow A$
$\neg(A \equiv B) \Leftarrow A \equiv B \Rightarrow \bot$
$\neg\neg A \Leftarrow A$

These clauses are not unique – there are many equivalent ways of writing the definition that gives the same truth function. We will show with one example that the truth function corresponding to this particular definition using the mapping $\mathcal{T}[]$ is indeed the same as the one given by the truth tables. The clauses for $\rightarrow$ give

$\mathcal{T}_{true}(X \rightarrow Y) = not\ not\ \mathcal{T}_{false}(X)\ or\ \mathcal{T}_{true}(Y) = \mathcal{T}_{false}(X)\ or\ \mathcal{T}_{true}(Y)$
$\mathcal{T}_{false}(X \rightarrow Y) = \mathcal{T}_{true}(X)\ and\ \mathcal{T}_{false}(Y)$

which is exactly the same as the truth function defined by the truth tables for *true* $\rightarrow$ and *false* $\rightarrow$.

The pseudo-interpretation to give logical truth is given by the clauses

$g(P) \Leftarrow g(P)$
$\neg g(P) \Leftarrow \neg g(P)$

Just as with the other representations of logics we have described, this pseudo-interpretation is incomplete. As in the other cases, we include *classical* in the antecedent to overcome the incompleteness – although "classical" is strictly speaking an improper name in this case, as there is no "intuitionistic" sense of the three-valued logics. A difference in this case is that we must not have a case where both *true*(*e*) and *false*(*e*) hold, as this does not correspond to a legal truth value. Instead, we must have three cases, corresponding to the legal combinations of the two judgments. We get

$classical \Leftarrow \Pi x\ classical'(x)$
$classical'(X) \Leftarrow X, \neg X \Rightarrow \bot$
$classical'(X) \Leftarrow \neg X, X \Rightarrow \bot$
$classical'(X) \Leftarrow X \Rightarrow \bot, \neg X \Rightarrow \bot$

$ctrue(A) \Leftarrow classical \Rightarrow A$

8 Related Work

The first presentation of a finitary deductive system based on the same ideas as those underlying the theory of partial inductive definitions was given by Hallnäs and Schroeder-Heister in [11]. That system did not include parameters or universal quantification (but did include logical variables). In [4], the present author used an informal finitary presentation of partial inductive definitions including parameters, but without unification of parameters in the $D\vdash$ rule. Instead, parameters were instantiated by a simple case analysis (this can be seen as a precursor of the ω-rule of [20]). Formalising the idea of case analysis and combining it with the $D\vdash$ rule resulted in the present finitary system, which was first presented in [5], and further developed in [6].

Another finitary presentation is that of Hanschke [13], using Skolemisation rather than parameters to represent universally quantified variables. Hanschke's approach is unsound in general (two different Skolem constants cannot be unified as two parameters can), but the unsound cases do not occur in the application for which the representation was intended (terminological reasoning).

Concerning general logics, our work is most similar to approaches where object logics are represented by expressing their inference rules using the metalogic. The most well-known work in this area are the Edinburgh Logical Framework (LF) [14] which represents logics using a dependent type system, and the metalogic of Isabelle [18] which uses higher-order resolution. A third interesting approach is that of Felty [7, 8].

Felty's work intends to specify theorem provers using a higher-order logic programming language (specifically λProlog [17]). In that respect, her approach is quite similar to that of Isabelle in that languages of the λProlog family implement higher-order resolution. There are, however, important differences in how the resolution mechanism is actually used. In particular, Felty's intention to specify theorem provers means that not only the inference rules themselves but also the derivation structures must be represented on the object level.

Since our metalogic (extended with logical variables as in [6]) can be regarded as a superset of the higher-order hereditary Harrop languages of the λProlog family, all of Felty's specifications can be implemented in our system.

Using our metalogic to implement Felty's specifications makes new interesting applications possible. With the power of a full sequent calculus system, including the $D\vdash$ rule, it would be possible to derive metalogical results about the specified deductive system. E.g. it should in principle be possible to derive such proof-theoretical results as cut-elimination or normalisation, although we have not made any practical attempts in this direction.

In [8], Felty also proves that LF can be interpreted in her system, so these approaches to general logic – as well as the Isabelle approach – have the same generality as ours. What is unique about our work is the semantic approach to general logic. The dependence of both the $\vdash D$ and $D\vdash$ rules on the definition, makes it possible to naturally interpret the definition as truth functions. Of course, the semantic approach could be emulated in other general logics – including LF – but that would not give the same natural correspondence between the inference rules of the general logic and the steps in evaluating the truth function.

As case examples, consider our representations of modal logic and three-valued logic in section 7. Both these representations have been carried out in LF as well [1]. However, the LF thesis, as given in [1]: "well-behaved natural deductive formalisms ... can be directly encoded" means that inference rules with side conditions generally force quite a complicated representation with much technical apparatus.

A case in point is the LF representation of natural deduction S4, which separates the system into a modal and nonmodal part with different validity judgments which are related in non-obvious ways. In contrast, our representation using Kripke semantics is quite natural. Modal logics are notorious for having complicated proof theories – in particular natural deduction systems – which makes them less well suited to a representation based on inference rules.

Benevides and Maibaum [2] have proposed a elegant and general natural deduction treatment of modal logic. It is interesting to see that although they do not use explicit accessibility relations, their treatment does make use of a concept resembling the "possible worlds" in the Kripke-semantic sense.

The representation of Kleene's three-valued logic in LF is essentially the same as ours. However the LF representation is motivated by a proof theory for the logic which was not present in the formulation of the logic [15]. While the LF representation requires the existence of a proof theory – if there is none, one would have to be specifically developed – our representation is directly based on the semantics, in this case the truth tables.

The back side of the coin is that the semantic approach requires a semantics for the logic which can be represented in a straightforward way. If the logic is given by inference rules only, the semantic approach can not be used.

9 Acknowledgements

I am indebted to the anonymous referee and to Roy Dyckhoff for suggestions on how to improve this paper.

10 References

1. Avron, A., Honsell, F., Mason, I.A., and Pollack, R.: Using Typed Lambda Calculus to Implement Formal Systems on a Machine, Journal of Automated Reasoning vol. 9 no. 3, 1992.
2. Benevides, M.R.F., and Maibaum, T.S.E.: A Constructive Presentation for the Modal Connective of Necessity, Journal of Logic and Computation vol. 2 no. 1, 1992.
3. Coquand, T. and Smith, J. M.: What is the status of pattern matching in type theory?, Proceedings of El Wintermöte, Programming Methodology Group Report PMG–R73, Department of Computer Sciences, Chalmers University of Technology, Gothenburg 1993.
4. Eriksson, L.-H. and Hallnäs, L.:, A Programming Calculus Based on Partial Inductive Definitions, SICS Research Report R88013, Swedish Institute of Computer Science, 1988.

5. Eriksson, L.-H.,: A Finitary Version of the Calculus of Partial Inductive Definitions, In: L.-H. Eriksson, L. Hallnäs and P. Schroeder-Heister (eds.): Extensions of Logic Programming - ELP '91, Lecture Notes in Artificial Intelligence 596, Springer-Verlag, 1992.
6. Eriksson, L.-H.: Finitary Partial Inductive Definitions and General Logic, Ph.D. thesis, Department of Computer and Systems Sciences, Royal Institute of Technology, Stockholm 1993.
7. Felty, A. and Miller, D.: Specifying Theorem Provers in a Higher-Order Logic Programming Language, 9th International Conference on Automated Deduction, Lecture Notes in Computer Science 310, Springer-Verlag, 1988.
8. Felty, A.P.: Specifying and Implementing Theorem Provers in a Higher-Order Logic Programming Language, Ph.D. thesis MS-CIS-89-53, University of Pennsylvania, 1989.
9. Gentzen, G.: Investigations into Logical Deduction (translation of Untersuchungen über das logische Schließen), In: Szabo, M.E. (ed.): The Collected Papers of Gerhard Gentzen, North-Holland, Amsterdam 1969.
10. Hallnäs, L.: On Normalisation of Proofs in Set Theory, Ph.D. thesis, Preprint no. 1 1983, Dept. of Philosophy, University of Stockholm, 1983.
11. Hallnäs, L. and Schroeder-Heister, P.: A Proof-Theoretical Approach to Logic Programming, Parts I and II, Journal of Logic and Computation, vol. 1, nos. 2 and 5, 1990 and 1991.
12. Hallnäs, L.: Partial Inductive Definitions, Theoretical Computer Science, vol. 87, no. 1, 1991.
13. Hanschke, P.: Terminological Reasoning and Partial Inductive Definitions, In: L.-H. Eriksson, L. Hallnäs and P. Schroeder-Heister (eds.): Extensions of Logic Programming – ELP '91, Lecture Notes in Artificial Intelligence 596, Springer-Verlag, 1992.
14. Harper, R., Honsell, F. and Plotkin, G.: A Framework for Defining Logics, Journal of the ACM, vol. 40, no. 1, 1993.
15. Kleene, S.C.: Introduction to Metamathematics, North-Holland, 1952.
16. Miller, D.: A Logic Programming Language with Lambda-Abstraction, Function Variables, and Simple Unification, In: P. Schroeder-Heister (ed.): Extensions of Logic Programming, Lecture Notes in Artificial Intelligence 475, Springer-Verlag, 1991.
17. Nadathur, G. and Miller, D.: An Overview of λProlog, Fifth International Conference of Logic Programming, pp. 810-827, MIT Press, 1988.
18. Paulson, L.C.: The Foundation of a Generic Theorem Prover, Journal of Automated Reasoning, vol. 5, no. 3, 1989.
19. Schroeder-Heister, P.: Structural Frameworks, Substructural Logics, and the Role of Elimination Inferences, In: Huet, G., and Plotkin, G. (eds.): Logical Frameworks, Cambridge University Press, 1991.
20. Schroeder-Heister, P.: Rules of Definitional Reflection, Proceedings of the Eighth Annual IEEE Symposium of Logic in Computer Science (LICS), pp. 222 – 232, 1993.