First Order Logic with Domain Conditions

F. Wiedijk and J. Zwanenburg

Department of Computer Science, University of Nijmegen
Toernooiveld 1, 6525 ED Nijmegen, The Netherlands

Abstract. This paper addresses the crucial issue in the design of a proof development system of how to deal with *partial functions* and the related question of how to treat *undefined terms*. Often the problem is avoided by artificially making all functions total. However, that does not correspond to the practice of everyday mathematics.
In type theory partial functions are modeled by giving functions extra arguments which are *proof objects*. In that case it will not be possible to apply functions outside their domain. However, having proofs as first class objects has the disadvantage that it will be unfamiliar to most mathematicians. Also many proof tools (like the theorem prover Otter) will not be usable for such a logic. Finally expressions get difficult to read because of these proof objects.
The PVS system solves the problem of partial functions differently. PVS generates *type-correctness conditions* (TCCs) for statements in its language. These are proof obligations that have to be satisfied 'on the side' to show that statements are well-formed.
We propose a TCC-like approach for the treatment of partial functions in type theory. We add *domain conditions* (DCs) to classical first-order logic and show the equivalence with a first order system that treats partial functions in the style of type theory.

1 Introduction

1.1 Problem

Until a few decades ago mathematics was something that was done in human heads, on the blackboard or on paper. Only since the seventies have systems been developed that verify mathematics with the computer. The first of these was the Automath system from the Netherlands. Other early systems of this kind were the Mizar system [10] from Poland and the LCF system from the UK. Recently this kind of system has become widely used (mostly because of applications in computer science). Currently the most popular is the PVS system [13] from a US company called SRI International. Other contemporary systems of this kind are ACL2 [8], IMPS [6] and NuPRL [2] from the US, HOL [5] and Isabelle [12] from the UK and Germany, and the Coq system [16] from France. This last system is an implementation of an approach to formalizing mathematics called *type theory*.

This paper addresses the question of how to treat partial functions in formal mathematics. The prototypical example of a partial function is division: the

D. Basin and B. Wolff (Eds.): TPHOLs 2003, LNCS 2758, pp. 221–237, 2003.

quotient $1/0$ is problematic because 0 is outside the domain of the division function. Formal systems have to take a position on how to deal with this kind of expression.

A traditional way to model partial functions in logic is by using *relations*. A statement about division is then interpreted as a statement about a ternary predicate $\mathsf{div\_eq}$, that satisfies the equivalence:

$$\mathsf{div\_eq}(x, y, z) \iff y \neq 0 \wedge x/y = z$$

However when translating statements this way, they become an order of magnitude larger than the original. Therefore, for actual implementations of formal systems it is not attractive.

In [7], John Harrison enumerated the four main approaches to partial functions that one actually encounters in proof checkers:

1. *Resolutely give each partial function a convenient value on points outside its domain.*
2. *Give each partial function some arbitrary value outside its domain.*
3. *Encode the domain of the partial function in its type and make its application to arguments outside that domain a type error.*
4. *Have a true logic of partial terms.*

In the first case one would define $1/0 = 0$, in the second case $1/0$ would be some real number but one would not be able to prove which one it is, in the third case $1/0$ would be a type error, and in the last case $1/0$ would be an allowed expression but it would not denote anything (and one would be able to prove so).

In the systems listed above, ACL2 uses the first approach, HOL, Isabelle and Mizar use a mixture of the first and second approaches, Coq, NuPRL and PVS use the third approach, and IMPS uses the fourth approach.

In this paper we explore a variant of Harrison's approach number 3. Although we do present a system of our own, it is not a 'logic of partial terms'. It does not allow one to write $1/0$ or any other undefined term and there is no way to state whether a term is defined (because it always is).

The approach that we present here is inspired by type theory, but our logic actually is one-sorted, so the variables of our logic all have the same 'type'. It is easy to generalize our approach to a many-sorted logic. We restricted ourselves to the one-sorted case for simplicity.

The problem that we address in this paper is how to be able to follow approach number 3, while still doing the proofs in the ordinary first order predicate logic with total functions.

There are two reasons why it is worthwhile not to have to give up first order logic:

- First order logic is the best known logic. Users of a proof checker will understand the system better if the logic is ordinary first order logic.

– There is much technology for first order logic. For instance there are many theorem provers for it. The best known of these is Otter [18], but there are many more. They even compete in first order theorem prover competitions like the CADE system competition. It is valuable to be able to use this technology in a proof checker.

1.2 Approach

We will introduce three logical systems, called system T, system D and system P. The names of those systems are abbreviations of 'total', 'domain' and 'partial'. These systems are:

System T. Ordinary first order logic with total functions.
System D. Exactly the same logic as system T, but in this system undefined expressions are not allowed. This means that all functions have to have the arguments inside their domain.
System P. A system in type theoretical style. Extra arguments which are proof objects ensure that it is not possible to write an undefined term.

Systems T and D have exactly the same set of expressions: only the derivations of both systems differ. System P has a different set of expressions, because it also has expressions for proof terms.

(For technical reasons we have an 'if-then-else' construction in all three systems. Therefore system T is not *really* ordinary first order logic, because it has something extra. However this if-then-else should not be considered to be an essential extension to the system. It should be considered 'syntactic sugar'. We do not treat the relation between the systems with and without the if-then-else in this paper. However we expect it to be unproblematic.)

Then for system T we introduce a notion of *domain conditions*. This is a set of proof obligations that has to be satisfied to ensure that functions are not applied outside their domain. For example the domain conditions of division are such that:

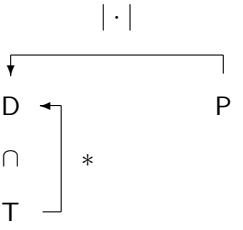
$$\mathcal{DC}(\frac{1}{x-1} - \frac{1}{x+1} = \frac{2}{x^2-1}) = \{\vdash x-1 \neq 0,\ \vdash x+1 \neq 0,\ \vdash x^2-1 \neq 0\}$$

Now the main theorem that we prove consists of Propositions 9 and 18 below. Together these state that:

A statement together with its domain conditions are provable in system T *iff that statement is provable in system* P.

This means that we can morally imagine ourselves as being in system P, while doing our proofs and the presentation of the statements in system T, as long as we also prove the domain conditions.

The relation between the three systems is outlined in the following diagram:



The $|\cdot|$ operation is called *erasure* (it is defined in Section 8). It erases all proof terms from the expressions. The $*$ operation is an auxiliary operation (as defined in Section 9) involved in the proof of the main theorem. It lifts a T proof to a D analogue.

1.3 Related Work

There are many logics of partial terms, like Scott's E-logic [14,17] and Beeson's LPT [1]. See [9] for an overview of the field. However our approach is not a logic of partial terms because we do not allow undefined terms. We think that $1/0$ should be *illegal*, and not just an undefined but legal expression.

Our paper integrates the type theoretical way to model partial functions with the PVS approach of having correctness conditions on the side. The type theoretical approach of having proof terms as an argument to model partial functions already dates from the Automath project (see [11] page 710). For a discussion of type-correctness conditions in PVS, see [15].

The approach that we propose here is similar to the one implemented in the LAMBDA system of Fourman [3], where each function f has an associated domain predicate `DOM'`f. However there is a difference in spirit: the LAMBDA system follows approach number 2 from the list on page 222,[1] while we follow approach number 3. This is also apparent from the fact that in [3] the $\mathcal{DC}$ operation (called $\ll\gg$ there) only appears in axioms corresponding to function definitions, while in our approach it takes a much more central position.

2 Examples

We will now list some examples of partial functions and show how they are treated in the systems T and P. Some of these functions occur in a proof of the fundamental theorem of algebra in the Coq system [4]. The experiences we had in this 'FTA project' was one of the motivations to write this paper.

– *Division.* In system T the division operator x/y gets a domain condition that has to be proved to show that the expression is well-formed:

$$\mathcal{DC}(x/y) = \{\vdash y \neq 0\}$$

[1] [3], p. 86: 'we regard a function application $f(x)$ as *always defined*, but if x is outside the intended domain of f, we will not be able to prove anything about its value.'

In system P division becomes a ternary function:

$$\mathsf{div}(x, y, \alpha)$$

having three arguments x, y and α, where α is a proof that $y \neq 0$.
This is an example of the general pattern. In system P all partial functions get one extra 'domain' argument. Therefore, in system P a function application $f(x_1, \ldots, x_n)$ becomes $f(x_1, \ldots, x_n, \alpha)$, where α is a proof of $D_f(x_1, \ldots, x_n)$ with the predicate D_f representing the *domain* of the function f.

- *Square root.* The domain condition of the real square root in system T is $\mathcal{DC}(\sqrt{x}) = \{\vdash x \geq 0\}$. Again, in system P the square root function $\mathsf{sqrt}(x, \alpha)$ has an extra argument, where α is a proof of $x \geq 0$.
- *Limit of a sequence.* The domain conditions of the limit operation are in system T:

$$DC\big(\lim_{n\to\infty} a_n\big) = \{\vdash \text{the sequence } (a_n) \text{ converges}\}$$

In system P the limit operation becomes $\mathsf{lim}(a, \alpha)$ where a is a sequence and where α is a proof that that sequence converges.
The natural way to express the limit operation in type theory is not first order, because the a argument is a function. But the theory we present here is first order. This means that this example should be considered in the context of set theory (like in the Mizar system) where one can talk about functions using a first order language.

- *Function application in set theory.* Set theory is untyped. When one defines function application in it, it becomes a partial operation. In set theory in the style of system T, function application would get the domain condition:

$$\mathcal{DC}(f(a)) = \{\vdash f \text{ is a function} \wedge a \in \mathsf{dom}\, f\}$$

Note that the dom function that occurs in this condition has a domain condition as well:

$$\mathcal{DC}(\mathsf{dom}\, f) = \{\vdash f \text{ is a relation}\}$$

In system P function application becomes ternary, $\mathsf{apply}(f, a, \alpha)$, where α is a proof of 'f is a function $\wedge\, a \in \mathsf{dom}\, f$'.

3 System T

We will now define the first of our three systems, namely first order predicate logic extended with an if-then-else operation.

To define our systems we fix a signature with finitely many constant, function and predicate symbols:

- constant symbols $c_1, \ldots, c_k$
- function symbols $f_1, \ldots, f_n$ with arities $a_1, \ldots, a_n$
- predicate symbols $P_1, \ldots, P_m$ with arities $r_1, \ldots, r_m$

We also have variables:

- term variables $x_0, x_1, x_2, \ldots$
- proof variables $h_0, h_1, h_2, \ldots$

(System T and D will only use term variables, but system P will also need proof variables.)

For each function symbol f of arity a there is a designated predicate symbol D_f (which is one of the P_i's) that also has arity a. D_f is the predicate that represents the *domain* of the function f. Note that this D_f is not an extra-logical abbreviation of a formula: it is a predicate *symbol*.

Constants could have been avoided by considering them to be nullary functions. However in that case our main result (Proposition 18 on page 235) would not have been true.[2]

System T has four kinds of expressions: terms, formulas, contexts and judgments. These expressions are defined inductively as the smallest sets $\mathcal{T}$, $\mathcal{F}$, $\mathcal{C}$ and $\mathcal{J}$ satisfying:

$$\begin{aligned}
\mathcal{T} &::= x_i \mid c_i \mid f_i(\mathcal{T}, \ldots, \mathcal{T}) \mid (\mathsf{if}\ \mathcal{F}\ \mathsf{then}\ \mathcal{T}\ \mathsf{else}\ \mathcal{T}) \\
\mathcal{F} &::= \bot \mid P_i(\mathcal{T}, \ldots, \mathcal{T}) \mid \mathcal{T} = \mathcal{T} \mid (\mathcal{F} \to \mathcal{F}) \mid (\forall x_i.\, \mathcal{F}) \\
\mathcal{C} &::= \epsilon \mid \mathcal{C},\, x_i \mid \mathcal{C},\, \mathcal{F} \\
\mathcal{J} &::= \mathcal{C} \vdash wf \mid \mathcal{C} \vdash \mathcal{T}\ wf \mid \mathcal{C} \vdash \mathcal{F}\ wf \mid \mathcal{C} \vdash \mathcal{F}
\end{aligned}$$

The four kinds of judgments mean respectively that the context is well-formed, that a term is well-formed, that a formula is well-formed and that a formula is provable.

Finally, system T has the following set of derivation rules:

$\mathcal{C}$:

$$(\epsilon\text{-}wf)\ \frac{}{\epsilon \vdash wf} \qquad (decl\text{-}wf)\ \frac{\Gamma \vdash wf}{\Gamma,\, x_i \vdash wf} \qquad (assum\text{-}wf)\ \frac{\Gamma \vdash \varphi\ wf}{\Gamma,\, \varphi \vdash wf}$$

$\mathcal{T}$:

$$(var\text{-}wf)\ \frac{\Gamma \vdash wf}{\Gamma \vdash x_i\ wf}\ x_i \in \Gamma \qquad (const\text{-}wf)\ \frac{\Gamma \vdash wf}{\Gamma \vdash c_i\ wf}$$

$$(fun\text{-}wf)\ \frac{\Gamma \vdash t_1\ wf \quad \cdots \quad \Gamma \vdash t_{a_i}\ wf \quad \Gamma \vdash wf}{\Gamma \vdash f_i(t_1, \ldots, t_{a_i})\ wf}$$

$$(if\text{-}wf)\ \frac{\Gamma \vdash \varphi\ wf \quad \Gamma \vdash t\ wf \quad \Gamma \vdash u\ wf}{\Gamma \vdash (\mathsf{if}\ \varphi\ \mathsf{then}\ t\ \mathsf{else}\ u)\ wf}$$

$\mathcal{F}$:

$$(\bot\text{-}wf)\ \frac{\Gamma \vdash wf}{\Gamma \vdash \bot\ wf} \qquad (pred\text{-}wf)\ \frac{\Gamma \vdash t_1\ wf \quad \cdots \quad \Gamma \vdash t_{r_i}\ wf \quad \Gamma \vdash wf}{\Gamma \vdash P_i(t_1, \ldots, t_{r_i})\ wf}$$

[2] This is shown by the following (pathological) case: one function symbol f of arity 0 with $D_f() \leftrightarrow \bot$ and $\varphi \equiv \exists x.\, \top$. This φ can be derived in the T system using witness $f()$, but it cannot be derived in the P system. This shows that it is essential for our theory to have constants c_i without domain conditions.

$$(\to\text{-}wf)\ \frac{\Gamma \vdash \varphi\ wf \quad \Gamma \vdash \psi\ wf}{\Gamma \vdash (\varphi \to \psi)\ wf} \qquad (\forall\text{-}wf)\ \frac{\Gamma,\, x_i \vdash \varphi\ wf}{\Gamma \vdash (\forall x_i.\, \varphi)\ wf}$$

$\mathcal{P}$:

$$(assum)\ \frac{\Gamma \vdash wf}{\Gamma \vdash \varphi}\ \varphi \in \Gamma \qquad (raa)\ \frac{\Gamma \vdash \neg\neg\varphi}{\Gamma \vdash \varphi}$$

$$(\to\text{-}I)\ \frac{\Gamma,\, \varphi \vdash \psi}{\Gamma \vdash (\varphi \to \psi)} \qquad (\to\text{-}E)\ \frac{\Gamma \vdash (\varphi \to \psi) \quad \Gamma \vdash \varphi}{\Gamma \vdash \psi}$$

$$(\forall\text{-}I)\ \frac{\Gamma,\, x_i \vdash \varphi}{\Gamma \vdash (\forall x_i.\, \varphi)} \qquad (\forall\text{-}E)\ \frac{\Gamma \vdash (\forall x_i.\, \varphi) \quad \Gamma \vdash t\ wf}{\Gamma \vdash \varphi[x_i := t]}$$

$$(refl)\ \frac{\Gamma \vdash t\ wf}{\Gamma \vdash t = t} \qquad (sym)\ \frac{\Gamma \vdash t = u}{\Gamma \vdash u = t} \qquad (trans)\ \frac{\Gamma \vdash t = u \quad \Gamma \vdash u = v}{\Gamma \vdash t = v}$$

$$(=\text{-}fun)\ \frac{\Gamma \vdash t_1 = t'_1 \quad \cdots \quad \Gamma \vdash t_{a_i} = t'_{a_i} \quad \Gamma \vdash wf}{\Gamma \vdash f_i(t_1, \ldots, t_{a_i}) = f_i(t'_1, \ldots, t'_{a_i})}$$

$$(=\text{-}pred)\ \frac{\Gamma \vdash t_1 = t'_1 \quad \cdots \quad \Gamma \vdash t_{r_i} = t'_{r_i} \quad \Gamma \vdash wf}{\Gamma \vdash P_i(t_1, \ldots, t_{r_i}) \to P_i(t'_1, \ldots, t'_{r_i})}$$

$$(=\text{-}if\text{-}true)\ \frac{\Gamma \vdash \varphi \quad \Gamma \vdash t\ wf \quad \Gamma \vdash u\ wf}{\Gamma \vdash (\mathsf{if}\ \varphi\ \mathsf{then}\ t\ \mathsf{else}\ u) = t}$$

$$(=\text{-}if\text{-}false)\ \frac{\Gamma \vdash \neg\varphi \quad \Gamma \vdash t\ wf \quad \Gamma \vdash u\ wf}{\Gamma \vdash (\mathsf{if}\ \varphi\ \mathsf{then}\ t\ \mathsf{else}\ u) = u}$$

We identify expressions that are α-equivalent. Therefore we assume in these rules the Barendregt convention: all variable names are as different as possible.

Logical operations have to be read as abbreviations from $\bot$, $\to$ and $\forall$:

$$\begin{aligned} \neg\varphi &\equiv \varphi \to \bot \\ \varphi \vee \psi &\equiv \neg\varphi \to \psi \\ \varphi \wedge \psi &\equiv \neg(\varphi \to \neg\psi) \\ \varphi \leftrightarrow \psi &\equiv (\varphi \to \psi) \wedge (\psi \to \varphi) \end{aligned}$$

Some remarks about system T:

- Our presentation of first order logic is slightly non-standard in that we have variables in the contexts that bind the free variables in the terms and formulas. Instead of '$x \neq 0 \vdash 1/x \neq 0$' we write '$x,\, x \neq 0 \vdash 1/x \neq 0$'. We do this for aesthetic reasons. It causes many well-formedness rules, but these are all trivial. In system T well-formedness just means that all free variables occur in the context.
- The condition $\Gamma \vdash wf$ in the rules *fun-wf*, *pred-wf*, *=-fun* and *=-pred* is needed for the case that $a_i = 0$ or $r_i = 0$.
- By symmetry of equality we can derive from rule *=-pred* the analogue rule with equivalence instead of implication:

$$(=\text{-}pred\text{-}iff)\ \frac{\Gamma \vdash t_1 = t'_1 \quad \cdots \quad \Gamma \vdash t_{r_i} = t'_{r_i} \quad \Gamma \vdash wf}{\Gamma \vdash P_i(t_1, \ldots, t_{r_i}) \leftrightarrow P_i(t'_1, \ldots, t'_{r_i})}$$

– It is possible to replace the rules *sym*, *trans*, *=-fun* and *=-pred* by just one *substitution* rule:

$$(=\text{-}subst)\ \frac{\Gamma \vdash t = u \quad \Gamma \vdash \varphi[x := t]}{\Gamma \vdash \varphi[x := u]}$$

However this rule does not generalize to systems D and P. The term t might satisfy domain conditions that u does not.

Our systems are non-standard because they have an if-then-else construction, which corresponds to the mathematical practice of definition by cases. For example one can write 'if $x \neq 0$ then $1/x$ else 0' as an expression. The if-then-else operator occurs in all three systems, T, D and P.

We would have preferred not to have this construction in our systems. In that case system T would really have been ordinary first order logic. However we need to have this construction to keep our definitions and proofs manageable. We have tried to develop the theory without it, but it became too complex.

We believe that the if-then-else is not essential to the systems. The theorem to be proved for this is conservativity of the extended system over the basic system:

> *If a judgment $\Gamma \vdash \varphi$ does not contain any* if-then-elses*, then it is provable in the system with* if-then-else *iff it is provable in the system without* if-then-else.

However we will not prove this theorem in this paper.

The if-then-elses can be eliminated systematically from a formula by replacing $P[(\mathsf{if}\ \varphi\ \mathsf{then}\ t\ \mathsf{else}\ u)]$ by $(\varphi \to P[t]) \wedge (\neg\varphi \to P[u])$. As an example $(\mathsf{if}\ x \neq 0\ \mathsf{then}\ 1/x\ \mathsf{else}\ 0) \cdot x = 1$ then becomes $(x \neq 0 \to (1/x) \cdot x = 1) \wedge (x = 0 \to 0 \cdot x = 1)$.

4 System D

The expressions of system D are exactly the same as the expressions of system T. Only the set of rules is different. We show the rules that differ from the corresponding rules in system T:

$\mathcal{T}$:
$$(\textit{fun-wf})\ \frac{\Gamma \vdash D_{f_i}(t_1, \ldots, t_{a_i})}{\Gamma \vdash f_i(t_1, \ldots, t_{a_i})\ wf} \qquad (\textit{if-wf})\ \frac{\Gamma, \varphi \vdash t\ wf \quad \Gamma, \neg\varphi \vdash u\ wf}{\Gamma \vdash (\mathsf{if}\ \varphi\ \mathsf{then}\ t\ \mathsf{else}\ u)\ wf}$$

$\mathcal{F}$:
$$(\to\text{-}wf)\ \frac{\Gamma, \varphi \vdash \psi\ wf}{\Gamma \vdash (\varphi \to \psi)\ wf}$$

$\mathcal{P}$:
$$(=\text{-}fun)\ \frac{\begin{array}{c}\Gamma \vdash t_1 = t'_1 \quad \cdots \quad \Gamma \vdash t_{a_i} = t'_{a_i} \\ \Gamma \vdash D_{f_i}(t_1, \ldots, t_{a_i}) \quad \Gamma \vdash D_{f_i}(t'_1, \ldots, t'_{a_i})\end{array}}{\Gamma \vdash f_i(t_1, \ldots, t_{a_i}) = f_i(t'_1, \ldots, t'_{a_i})}$$

$$(=\text{-}\mathit{if\text{-}true})\ \frac{\Gamma \vdash \varphi \quad \Gamma, \varphi \vdash t\ wf \quad \Gamma, \neg\varphi \vdash u\ wf}{\Gamma \vdash (\mathsf{if}\ \varphi\ \mathsf{then}\ t\ \mathsf{else}\ u) = t}$$

$$(=\text{-}\mathit{if\text{-}false})\ \frac{\Gamma \vdash \neg\varphi \quad \Gamma, \varphi \vdash t\ wf \quad \Gamma, \neg\varphi \vdash u\ wf}{\Gamma \vdash (\mathsf{if}\ \varphi\ \mathsf{then}\ t\ \mathsf{else}\ u) = u}$$

The essential differences between systems T and D are the rules *fun-wf* and *=-fun*. In system D you are only allowed to apply a function if you can prove that the arguments are in its domain.

The other differences are not essential: rules *if-wf*, *→-wf*, *=-if-true* and *=-if-false* in system T could have been the same as in system D. (But not the other way around: in system D these rules have to be the way they are.) However we have chosen to use in system T the simpler variants of those rules. The slight differences between systems T and D in this respect do not cause any problems in the proofs below.

5 System P

The expressions of system P follow the same basic structure as the expressions of system T and D. However there is an extra kind of expression $\mathcal{P}$, for proof terms.

$$\begin{aligned}
\mathcal{T} &::= x_i \mid c_i \mid f_i(\mathcal{T}, \ldots, \mathcal{T}, \mathcal{P}) \mid (\mathsf{if}\ \mathcal{F}\ \mathsf{then}\ \lambda h_i.\mathcal{T}\ \mathsf{else}\ \lambda h_j.\mathcal{T}) \\
\mathcal{F} &::= \bot \mid P_i(\mathcal{T}, \ldots, \mathcal{T}) \mid \mathcal{T} = \mathcal{T} \mid (\Pi h_i : \mathcal{F}.\mathcal{F}) \mid (\forall x_i.\mathcal{F}) \\
\mathcal{P} &::= (\lambda h_i : \mathcal{F}.\mathcal{P}) \mid (\mathcal{P}\mathcal{P}) \mid (\lambda x_i.\mathcal{P}) \mid (\mathcal{P}\mathcal{T}) \\
&\quad \mid \mathsf{raa}(\mathcal{P}) \mid \mathsf{refl}(\mathcal{T}) \mid \mathsf{sym}(\mathcal{P}) \mid \mathsf{trans}(\mathcal{P}, \mathcal{P}) \\
&\quad \mid \mathsf{eqfun}(\mathcal{P}, \ldots \mathcal{P}, \mathcal{P}, \mathcal{P}) \mid \mathsf{eqpred}(i, \mathcal{P}, \ldots \mathcal{P}) \\
&\quad \mid \mathsf{iftrue}(\mathcal{P}, \lambda h_i.\mathcal{T}, \lambda h_j.\mathcal{T}) \mid \mathsf{iffalse}(\mathcal{P}, \lambda h_i.\mathcal{T}, \lambda h_j.\mathcal{T}) \\
\mathcal{C} &::= \epsilon \mid \mathcal{C},\ x_i \mid \mathcal{C},\ h_i : \mathcal{F} \\
\mathcal{J} &::= \mathcal{C} \vdash wf \mid \mathcal{C} \vdash \mathcal{T}\ wf \mid \mathcal{C} \vdash \mathcal{F}\ wf \mid \mathcal{C} \vdash \mathcal{P} : \mathcal{F}
\end{aligned}$$

Here are the rules for system P. They exactly parallel the rules for system D.

$\mathcal{C}$: $\quad (\epsilon\text{-}\mathit{wf})\ \dfrac{}{\epsilon \vdash wf} \quad (\mathit{decl\text{-}wf})\ \dfrac{\Gamma \vdash wf}{\Gamma,\ x_i \vdash wf} \quad (\mathit{assum\text{-}wf})\ \dfrac{\Gamma \vdash \varphi\ wf}{\Gamma,\ h_i : \varphi \vdash wf}$

$\mathcal{T}$: $\quad (\mathit{var\text{-}wf})\ \dfrac{\Gamma \vdash wf}{\Gamma \vdash x_i\ wf}\ x_i \in \Gamma \quad (\mathit{const\text{-}wf})\ \dfrac{\Gamma \vdash wf}{\Gamma \vdash c_i\ wf}$

$$(\mathit{fun\text{-}wf})\ \frac{\Gamma \vdash \alpha : D_{f_i}(t_1, \ldots, t_{a_i})}{\Gamma \vdash f_i(t_1, \ldots, t_{a_i}, \alpha)\ wf} \quad (\mathit{if\text{-}wf})\ \frac{\Gamma,\ h_i : \varphi \vdash t\ wf \quad \Gamma,\ h_j : \neg\varphi \vdash u\ wf}{\Gamma \vdash (\mathsf{if}\ \varphi\ \mathsf{then}\ \lambda h_i.t\ \mathsf{else}\ \lambda h_j.u)\ wf}$$

$\mathcal{F}$: $\quad (\bot\text{-}\mathit{wf})\ \dfrac{\Gamma \vdash wf}{\Gamma \vdash \bot\ wf} \quad (\mathit{pred\text{-}wf})\ \dfrac{\Gamma \vdash t_1\ wf \quad \cdots \quad \Gamma \vdash t_{r_i}\ wf \quad \Gamma \vdash wf}{\Gamma \vdash P_i(t_1, \ldots, t_{r_i})\ wf}$

$$(\Pi\text{-}wf)\ \frac{\Gamma,\, h_i : \varphi \vdash \psi\ wf}{\Gamma \vdash (\Pi h_i : \varphi.\, \psi)\ wf} \quad (\forall\text{-}wf)\ \frac{\Gamma,\, x_i \vdash \varphi\ wf}{\Gamma \vdash (\forall x_i.\, \varphi)\ wf}$$

$\mathcal{P}$:

$$(assum)\ \frac{\Gamma \vdash wf}{\Gamma \vdash h_i : \varphi}\ h_i : \varphi \in \Gamma \quad (raa)\ \frac{\Gamma \vdash \alpha : \neg\neg\varphi}{\Gamma \vdash \mathsf{raa}(\alpha) : \varphi}$$

$$(\Pi\text{-}I)\ \frac{\Gamma,\, h_i : \varphi \vdash \alpha : \psi}{\Gamma \vdash (\lambda h_i : \varphi.\, \alpha) : (\Pi h_i : \varphi.\, \psi)} \quad (\Pi\text{-}E)\ \frac{\Gamma \vdash \alpha : (\Pi h_i : \varphi.\, \psi) \quad \Gamma \vdash \beta : \varphi}{\Gamma \vdash (\alpha\beta) : \psi[h_i := \beta]}$$

$$(\forall\text{-}I)\ \frac{\Gamma,\, x_i \vdash \alpha : \varphi}{\Gamma \vdash (\lambda x_i.\, \alpha) : (\forall x_i.\, \varphi)} \quad (\forall\text{-}E)\ \frac{\Gamma \vdash \alpha : (\forall x_i.\, \varphi) \quad \Gamma \vdash t\ wf}{\Gamma \vdash (\alpha t) : \varphi[x_i := t]}$$

$$(refl)\ \frac{\Gamma \vdash t\ wf}{\Gamma \vdash \mathsf{refl}(t) : t = t} \quad (sym)\ \frac{\Gamma \vdash \alpha : t = u}{\Gamma \vdash \mathsf{sym}(\alpha) : u = t}$$

$$(trans)\ \frac{\Gamma \vdash \alpha : t = u \quad \Gamma \vdash \beta : u = v}{\Gamma \vdash \mathsf{trans}(\alpha, \beta) : t = v}$$

$$(=\text{-}fun)\ \frac{\begin{array}{c}\Gamma \vdash \alpha_1 : t_1 = t'_1 \quad \cdots \quad \Gamma \vdash \alpha_{a_i} : t_{a_i} = t'_{a_i} \\ \Gamma \vdash \beta : D_{f_i}(t_1, \ldots, t_{a_i}) \quad \Gamma \vdash \beta' : D_{f_i}(t'_1, \ldots, t'_{a_i})\end{array}}{\Gamma \vdash \mathsf{eqfun}(\alpha_1, \ldots \alpha_{a_i}, \beta, \beta') : f_i(t_1, \ldots, t_{a_i}, \beta) = f_i(t'_1, \ldots, t'_{a_i}, \beta')}$$

$$(=\text{-}pred)\ \frac{\Gamma \vdash \alpha_1 : t_1 = t'_1 \quad \cdots \quad \Gamma \vdash \alpha_{r_i} : t_{r_i} = t'_{r_i} \quad \Gamma \vdash wf}{\Gamma \vdash \mathsf{eqpred}(i, \alpha_1, \ldots \alpha_{r_i}) : P_i(t_1, \ldots, t_{r_i}) \to P_i(t'_1, \ldots, t'_{r_i})}$$

$$(=\text{-}if\text{-}true)\ \frac{\Gamma \vdash \alpha : \varphi \quad \Gamma,\, h_i : \varphi \vdash t\ wf \quad \Gamma,\, h_j : \neg\varphi \vdash u\ wf}{\Gamma \vdash \mathsf{iftrue}(\alpha, \lambda h_i.\, t, \lambda h_j.\, u) : (\mathsf{if}\ \varphi\ \mathsf{then}\ \lambda h_i.\, t\ \mathsf{else}\ \lambda h_j.\, u) = t[h_i := \alpha]}$$

$$(=\text{-}if\text{-}false)\ \frac{\Gamma \vdash \alpha : \neg\varphi \quad \Gamma,\, h_i : \varphi \vdash t\ wf \quad \Gamma,\, h_j : \neg\varphi \vdash u\ wf}{\Gamma \vdash \mathsf{iffalse}(\alpha, \lambda h_i.\, t, \lambda h_j.\, u) : (\mathsf{if}\ \varphi\ \mathsf{then}\ \lambda h_i.\, t\ \mathsf{else}\ \lambda h_j.\, u) = u[h_j := \alpha]}$$

We will write $\Gamma \vdash \varphi$ if for some α we can derive that $\Gamma \vdash \alpha : \varphi$. If h_i does not occur in ψ we write $\varphi \to \psi$ for $\Pi h_i : \varphi.\, \psi$ like before.

The expressions for the if-then-else bind two proof variables: one for the then branch and one for the else branch. This is indicated by putting λs in the if-then-else, iffalse and iftrue expressions. These λs should not be confused with the λ-expressions that occur in the proof terms of an implication or universally quantified formula, which are introduced by the Π-I and $\forall$-I rules.

System P does not have a conversion rule. Without a conversion rule the system does not satisfy the property of subject reduction. We do not think a conversion rule is relevant for our application. We expect that adding a conversion rule will not affect the results from this paper.

Proof terms only occur as the final arguments of functions.

6 Some Properties of Derivations

The systems T, D and P are well behaved. All three systems satisfy the following four propositions (where $\mathcal{X}$ is anything that can occur after a $\vdash$):

Proposition 1. *$\Gamma, \Gamma' \vdash \mathcal{X}$ then with a shorter derivation $\Gamma \vdash wf$*

Proposition 2. *$\Gamma, \varphi, \Gamma' \vdash \mathcal{X}$ then with a shorter derivation $\Gamma \vdash \varphi$ wf*

Proposition 3. *$\Gamma \vdash \varphi$ implies $\Gamma \vdash \varphi$ wf.*

Proposition 4 (weakening). *$\Gamma \vdash \mathcal{X}$ and $\Gamma, \Gamma' \vdash$ wf imply $\Gamma, \Gamma' \vdash \mathcal{X}$.*

7 The Domain Conditions

We now define the domain conditions of an expression. For each system T expression (term, formula, judgment), its domain conditions are a set of judgments that state that in the expression no function is applied outside its domain.

Domain conditions are defined relative to a context Γ which we put as a subscript to the $\mathcal{DC}$ symbol.

$$
\begin{array}{l}
\mathcal{DC}_\Gamma : \mathcal{T}^{\mathsf{T}} \to \mathcal{P}(\mathcal{J}^{\mathsf{T}}) \\
\quad \mathcal{DC}_\Gamma(x_i) = \mathcal{DC}_\Gamma(c_i) = \emptyset \\
\quad \mathcal{DC}_\Gamma(f_i(t_1, \ldots, t_{a_i})) = \mathcal{DC}_\Gamma(t_1) \cup \ldots \cup \mathcal{DC}_\Gamma(t_{a_i}) \cup \{\Gamma \vdash^{\mathsf{T}} D_{f_i}(t_1, \ldots, t_{a_i})\} \\
\quad \mathcal{DC}_\Gamma(\mathsf{if}\ \varphi\ \mathsf{then}\ t\ \mathsf{else}\ u) = \mathcal{DC}_\Gamma(\varphi) \cup \mathcal{DC}_{\Gamma,\varphi}(t) \cup \mathcal{DC}_{\Gamma,\neg\varphi}(u) \\
\mathcal{DC}_\Gamma : \mathcal{F}^{\mathsf{T}} \to \mathcal{P}(\mathcal{J}^{\mathsf{T}}) \\
\quad \mathcal{DC}_\Gamma(\bot) = \emptyset \\
\quad \mathcal{DC}_\Gamma(P_i(t_1, \ldots, t_{r_i})) = \mathcal{DC}_\Gamma(t_1) \cup \ldots \cup \mathcal{DC}_\Gamma(t_{r_i}) \\
\quad \mathcal{DC}_\Gamma(t = u) = \mathcal{DC}(t) \cup \mathcal{DC}(u) \\
\quad \mathcal{DC}_\Gamma(\varphi \to \psi) = \mathcal{DC}_\Gamma(\varphi) \cup \mathcal{DC}_{\Gamma,\varphi}(\psi) \\
\quad \mathcal{DC}_\Gamma(\forall x_i.\, \varphi) = \mathcal{DC}_{\Gamma,x_i}(\varphi) \\
\mathcal{DC} : \mathcal{C}^{\mathsf{T}} \to \mathcal{P}(\mathcal{J}^{\mathsf{T}}) \\
\quad \mathcal{DC}(\epsilon) = \emptyset \\
\quad \mathcal{DC}(\Gamma, x_i) = \mathcal{DC}(\Gamma) \\
\quad \mathcal{DC}(\Gamma, \varphi) = \mathcal{DC}(\Gamma) \cup \mathcal{DC}_\Gamma(\varphi)
\end{array}
$$

Domain conditions are asymmetric in some of the propositional connectives. Therefore, although system T is just first order logic, the domain conditions do not respect logical equivalence. For instance:

$$\mathcal{DC}(\varphi \wedge \psi) \neq \mathcal{DC}(\psi \wedge \varphi)$$

In the first case $\varphi \wedge \psi \equiv \neg(\varphi \to \neg\psi)$ and we can use φ for proving the domain conditions of ψ, while in the second case $\psi \wedge \varphi \equiv \neg(\psi \to \neg\varphi)$ so in that case we can *not* use φ for the domain conditions of ψ. As an example the domain conditions of $(x \neq 0) \wedge P[1/x]$ might be provable because we can use $x \neq 0$ to prove the domain conditions of $P[1/x]$, but the domain conditions of $P[1/x] \wedge (x \neq 0)$ might not be provable because in that case we have to prove the domain conditions of $P[1/x]$ without the benefit of $x \neq 0$. In this sense the $\wedge$ connective in system T behaves like the `&&` operator of the C programming language.

The predicate symbol D_f is a *symbol* that *represents* the domain of the function f. To give this symbol a meaning we have to have an equivalence in the context.

For instance $D_/(x, y)$ is the definedness predicate of the division. It should be equivalent to $y \neq 0$. This means that we have to imagine that we are reasoning in a context:

$$\Gamma \equiv \ldots, \textit{ theory of division including } \forall x, y.\, (D_/(x, y) \leftrightarrow y \neq 0), \ldots$$

The domain condition for division is: $\mathcal{DC}(x/y) = \{\vdash D_/(x, y)\}$. This domain condition is equivalent to $y \neq 0$ in the proper context but it is not identical to it. (So actually in the examples on pages 223–225 we were not completely correct. We 'cheated' there for the sake of the presentation.)

8 The Erasure: From P to T

The erasure operation $|\cdot|$ erases all proof terms from expressions. It maps system P expressions to system T. In such an erased expression all domain conditions hold. This is the easy direction of our main result.

$|\cdot| : \mathcal{T}^{\mathsf{P}} \to \mathcal{T}^{\mathsf{T}}$
$\quad |x_i| = x_i \quad |c_i| = c_i$
$\quad |f_i(t_1, \ldots, t_{a_i}, \alpha)| = f_i(|t_1|, \ldots, |t_{a_i}|)$
$\quad |(\mathsf{if}\ \varphi\ \mathsf{then}\ \lambda h_i.\, t\ \mathsf{else}\ \lambda h_j.\, u)| = (\mathsf{if}\ |\varphi|\ \mathsf{then}\ |t|\ \mathsf{else}\ |u|)$
$|\cdot| : \mathcal{F}^{\mathsf{P}} \to \mathcal{F}^{\mathsf{T}}$
$\quad |\bot| = \bot$
$\quad |P_i(t_1, \ldots, t_{r_i})| = P_i(|t_1|, \ldots, |t_{r_i}|)$
$\quad |t = u| = |t| = |u|$
$\quad |(\Pi h_i : \varphi.\, \psi)| = (|\varphi| \to |\psi|)$
$\quad |(\forall x_i.\, \varphi)| = (\forall x_i.\, |\varphi|)$
$|\cdot| : \mathcal{C}^{\mathsf{P}} \to \mathcal{C}^{\mathsf{T}}$
$\quad |\epsilon| = \epsilon$
$\quad |\Gamma,\, x_i| = |\Gamma|,\, x_i$
$\quad |\Gamma,\, h_i : \varphi| = |\Gamma|,\, |\varphi|$

Proposition 5.

$|t[x_i := u]| \equiv |t|[x_i := |u|]$.
$|\varphi[x_i := u]| \equiv |\varphi|[x_i := |u|]$.
$|t[h_i := \alpha]| \equiv |t|$.
$|\varphi[h_i := \alpha]| \equiv |\varphi|$.

Proof. Simultaneous induction on the structure of t and φ.

Proposition 6 (from P to D).

$\Gamma \vdash^{\mathsf{P}} \textit{wf}$ *implies* $|\Gamma| \vdash^{\mathsf{D}} \textit{wf}$.
$\Gamma \vdash^{\mathsf{P}} t\ \textit{wf}$ *implies* $|\Gamma| \vdash^{\mathsf{D}} |t|\ \textit{wf}$.
$\Gamma \vdash^{\mathsf{P}} \varphi\ \textit{wf}$ *implies* $|\Gamma| \vdash^{\mathsf{D}} |\varphi|\ \textit{wf}$.

$\Gamma \vdash^{\mathsf{P}} \varphi$ *implies* $|\Gamma| \vdash^{\mathsf{D}} |\varphi|$.

Proof. Simultaneous induction on the size of the derivation in system P.

Proposition 7.

$\Gamma, \varphi \vdash^{\mathsf{T}} t\ wf$ *implies* $\Gamma \vdash^{\mathsf{T}} t\ wf$.
$\Gamma, \varphi \vdash^{\mathsf{T}} \psi\ wf$ *implies* $\Gamma \vdash^{\mathsf{T}} \psi\ wf$.

Proof. Well-formedness in system T just checks whether all free variables are in the context. Removing assumptions from the context does not affect that.

Proposition 8 (from D to T).

$\Gamma \vdash^{\mathsf{D}} wf$ *implies* $\Gamma \vdash^{\mathsf{T}} wf$ *and* $\mathcal{DC}(\Gamma)$.
$\Gamma \vdash^{\mathsf{D}} t\ wf$ *implies* $\Gamma \vdash^{\mathsf{T}} t\ wf$ *and* $\mathcal{DC}(\Gamma)$ *and* $\mathcal{DC}_\Gamma(t)$.
$\Gamma \vdash^{\mathsf{D}} \varphi\ wf$ *implies* $\Gamma \vdash^{\mathsf{T}} \varphi\ wf$ *and* $\mathcal{DC}(\Gamma)$ *and* $\mathcal{DC}_\Gamma(\varphi)$.
$\Gamma \vdash^{\mathsf{D}} \varphi$ *implies* $\Gamma \vdash^{\mathsf{T}} \varphi$ *and* $\mathcal{DC}(\Gamma)$ *and* $\mathcal{DC}_\Gamma(\varphi)$.

Proof. Simultaneous induction on the size of the derivation in system D using Proposition 7 for the $\rightarrow$*-wf*, *if-wf*, $=$*-if-true* and $=$*-if-false* rules.

Proposition 9 (main correspondence theorem from P to T). $\Gamma \vdash^{\mathsf{P}} \varphi$ *implies* $|\Gamma| \vdash^{\mathsf{T}} |\varphi|$ *and* $\mathcal{DC}(|\Gamma|)$ *and* $\mathcal{DC}_{|\Gamma|}(|\varphi|)$.

Proof. Propositions 6 and 8 combined.

9 The * Operation: From T to D

Consider a statement for which the domain conditions hold. A system T proof of this statement and a system D proof of the same statement are different things. In the first case, although the domain conditions of the statement are satisfied, the *proof* might violate some domain conditions (for instance, it might reason about $1/0$ as a number). But in the second case the domain conditions have to hold *in all steps* of the proof. We will show that despite this difference these two kinds of provability are equivalent (this is Proposition 15 below). For this we will use the $*$ operation.

The $*$ operation maps system T to system D. It makes the partial functions total by setting them to the constant c_1 outside their domain. Then system T proofs are interpreted in system D as talking about these 'extended' functions.

$\cdot^* : \mathcal{T}^{\mathsf{T}} \to \mathcal{T}^{\mathsf{D}}$
$x_i{}^* = x_i \quad c_i{}^* = c_i$
$f_i(t_1, \ldots, t_{a_i})^* = (\mathsf{if}\ D_{f_i}(t_1{}^*, \ldots, t_{a_i}{}^*)\ \mathsf{then}\ f_i(t_1{}^*, \ldots, t_{a_i}{}^*)\ \mathsf{else}\ c_1)$
$(\mathsf{if}\ \varphi\ \mathsf{then}\ t\ \mathsf{else}\ u)^* = (\mathsf{if}\ \varphi^*\ \mathsf{then}\ t^*\ \mathsf{else}\ u^*)$
$\cdot^* : \mathcal{F}^{\mathsf{T}} \to \mathcal{F}^{\mathsf{D}}$

$$
\begin{aligned}
&\bot^* = \bot \\
&P_i(t_1, \ldots, t_{r_i})^* = P_i({t_1}^*, \ldots, {t_{r_i}}^*) \\
&(t = u)^* = t^* = u^* \\
&(\varphi \to \psi)^* = (\varphi^* \to \psi^*) \\
&(\forall x_i.\, \varphi)^* = (\forall x_i.\, \varphi^*) \\
\cdot^* : \mathcal{C}^{\mathsf{T}} &\to \mathcal{C}^{\mathsf{D}} \\
&\epsilon^* = \epsilon \\
&(\Gamma,\, x_i)^* = \Gamma^*,\, x_i \\
&(\Gamma,\, \varphi)^* = \Gamma^*,\, \varphi^*
\end{aligned}
$$

Proposition 10.

$\Gamma \vdash^{\mathsf{D}} t\ wf$ *implies* $\Gamma \vdash^{\mathsf{D}} t = t^*$.
$\Gamma \vdash^{\mathsf{D}} \varphi\ wf$ *implies* $\Gamma \vdash^{\mathsf{D}} \varphi \leftrightarrow \varphi^*$.

Proof. Simultaneous induction on the structure of t and φ.

Proposition 11. $\Gamma \vdash^{\mathsf{D}} wf$ *implies that* $\Gamma \vdash^{\mathsf{D}} \varphi$ *iff* $\Gamma^* \vdash^{\mathsf{D}} \varphi$.

Proof. Induction on the structure of Γ using the second part of Proposition 10.

Proposition 12.

$\Gamma \vdash^{\mathsf{T}} wf$ *implies* $\Gamma^* \vdash^{\mathsf{D}} wf$.
$\Gamma \vdash^{\mathsf{T}} t\ wf$ *implies* $\Gamma^* \vdash^{\mathsf{D}} t^*\ wf$.
$\Gamma \vdash^{\mathsf{T}} \varphi\ wf$ *implies* $\Gamma^* \vdash^{\mathsf{D}} \varphi^*\ wf$.
$\Gamma \vdash^{\mathsf{T}} \varphi$ *implies* $\Gamma^* \vdash^{\mathsf{D}} \varphi^*$.

Proof. Simultaneous induction on the size of the derivation in system T using the previous two propositions.

Proposition 13.

$\Gamma \vdash^{\mathsf{D}} wf$ *and* $\mathcal{DC}_\Gamma(t)$ *imply* $\Gamma \vdash^{\mathsf{D}} t = t^*$.
$\Gamma \vdash^{\mathsf{D}} wf$ *and* $\mathcal{DC}_\Gamma(\varphi)$ *imply* $\Gamma \vdash^{\mathsf{D}} \varphi \leftrightarrow \varphi^*$.

Proof. Simultaneous induction on the structure of t and φ.

Proposition 14. $\mathcal{DC}(\Gamma)$ *implies that* $\Gamma \vdash^{\mathsf{D}} \varphi$ *iff* $\Gamma^* \vdash^{\mathsf{D}} \varphi$.

Proof. Induction on the structure of Γ using the second part of Proposition 13.

Proposition 15. $\mathcal{DC}(\Gamma)$ *and* $\mathcal{DC}_\Gamma(\varphi)$ *and* $\Gamma \vdash^{\mathsf{T}} \varphi$ *imply* $\Gamma \vdash^{\mathsf{D}} \varphi$.

Proof. Assume $\mathcal{DC}(\Gamma)$, $\mathcal{DC}_\Gamma(\varphi)$ and $\Gamma \vdash^{\mathsf{T}} \varphi$. Then $\Gamma^* \vdash^{\mathsf{D}} \varphi^*$ by Proposition 12 and then $\Gamma \vdash^{\mathsf{D}} \varphi^*$ by Proposition 14 and therefore $\Gamma \vdash^{\mathsf{D}} \varphi$ by the second part of Proposition 13.

10 From D to P

In the previous section we got from T to D. Now we show how to get from D to P. To 'fill in' the proof objects in a system D proof we need a property called *proof irrelevance*. It says that a P expression does not change its meaning if we replace proof terms in it with different proofs of the same statements. This is stated 'locally' in the *=-fun* rule of system P:

$$\frac{\Gamma \vdash t_1 = t'_1 \quad \cdots \quad \Gamma \vdash t_{a_i} = t'_{a_i} \qquad \Gamma \vdash \beta : D_{f_i}(t_1, \ldots, t_{a_i}) \quad \Gamma \vdash \beta' : D_{f_i}(t'_1, \ldots, t'_{a_i})}{\Gamma \vdash f_i(t_1, \ldots, t_{a_i}, \beta) = f_i(t'_1, \ldots, t'_{a_i}, \beta')}$$

The two terms are equal despite the proof terms β and β' being different. But the property of proof irrelevance is also true 'globally':

Proposition 16 (proof irrelevance).

$\Gamma \vdash^{\mathsf{P}} t\ wf$ and $\Gamma \vdash^{\mathsf{P}} t'\ wf$ and $|t| \equiv |t'|$ imply $\Gamma \vdash^{\mathsf{P}} t = t'$.
$\Gamma \vdash^{\mathsf{P}} \varphi\ wf$ and $\Gamma \vdash^{\mathsf{P}} \varphi'\ wf$ and $|\varphi| \equiv |\varphi'|$ imply $\Gamma \vdash^{\mathsf{P}} \varphi \leftrightarrow \varphi'$.

Proof. Simultaneous induction on the size of $|t|$ and $|\varphi|$ using Proposition 5 and the *=-fun*, *=-pred*, *=-if-true* and *=-if-false* rules.

Once we have proof irrelevance, getting a system P derivation from a system D derivation is straightforward. Together with the earlier propositions this then allows us to prove the main result of this paper.

Proposition 17. *If $\Gamma \vdash^{\mathsf{P}} wf$ then:*

$|\Gamma| \vdash^{\mathsf{D}} t'\ wf$ imply that there exists a t with $|t| \equiv t'$ such that $\Gamma \vdash^{\mathsf{P}} t\ wf$.
$|\Gamma| \vdash^{\mathsf{D}} \varphi'\ wf$ imply that there exists a φ with $|\varphi| \equiv \varphi'$ such that $\Gamma \vdash^{\mathsf{P}} \varphi\ wf$.
$|\Gamma| \vdash^{\mathsf{D}} \varphi'$ imply that there exists a φ with $|\varphi| \equiv \varphi'$ such that $\Gamma \vdash^{\mathsf{P}} \varphi$.

Proof. Simultaneous induction on the size of the derivation in system D using Propositions 5 and 16.

Proposition 18 (main correspondence theorem from T to P). *$\Gamma \vdash^{\mathsf{P}} wf$ and $|\Gamma| \vdash^{\mathsf{T}} \varphi'$ and $\mathcal{DC}_{|\Gamma|}(\varphi')$ imply that there exists a φ with $|\varphi| \equiv \varphi'$ such that $\Gamma \vdash^{\mathsf{P}} \varphi$.*

Proof. Assume $\Gamma \vdash^{\mathsf{P}} wf$, $|\Gamma| \vdash^{\mathsf{T}} \varphi'$ and $\mathcal{DC}_{|\Gamma|}(\varphi')$. Then $\mathcal{DC}(|\Gamma|)$ by Proposition 9 and then $|\Gamma| \vdash^{\mathsf{D}} \varphi'$ by Proposition 15 and therefore there exists a suitable φ by Proposition 17.

Proposition 19 (corollary). *$\Gamma' \vdash^{\mathsf{T}} \varphi'$ and $\mathcal{DC}(\Gamma')$ and $\mathcal{DC}_{\Gamma'}(\varphi')$ imply that there exist Γ and φ with $|\Gamma| \equiv \Gamma'$ and $|\varphi| \equiv \varphi'$ such that $\Gamma \vdash^{\mathsf{P}} \varphi$.*

Proof. $\Gamma' \vdash^{\mathsf{D}} \varphi'$ by Proposition 15, from which $\Gamma' \vdash^{\mathsf{D}} wf$. Then $\Gamma' \vdash^{\mathsf{D}} wf$ implies that there exists a Γ with $|\Gamma| \equiv \Gamma'$ such that $\Gamma \vdash^{\mathsf{P}} wf$, by induction on the structure of Γ' using Proposition 17. Finally we get φ from Proposition 18.

11 Conclusion

The main things left to be done are:

1. Prove the systems with if-then-else conservative over the same systems without this construction.
2. Build a proof assistant that implements the approach of reasoning in system T with domain conditions, to study how well it works in practice.
3. Investigate whether the theory from this paper extends to higher order logic.

Acknowledgments. Thanks to Herman Geuvers, Paula Severi and Venanzio Capretta for stimulating discussions. Thanks to Gilles Barthe for the idea of how to fit this paper in 16 pages. Thanks to the anonymous referees for valuable comments.

References

1. M.J. Beeson. *Foundations of constructive mathematics*. Springer-Verlag, 1985.
2. Robert L. Constable, Stuart F. Allen, H.M. Bromley, W.R. Cleaveland, J.F. Cremer, R.W. Harper, Douglas J. Howe, T.B. Knoblock, N.P. Mendler, P. Panangaden, James T. Sasaki, and Scott F. Smith. *Implementing Mathematics with the Nuprl Development System*. Prentice-Hall, NJ, 1986.
3. Simon Finn, Michael Fourman, and John Longley. Partial Functions in a Total Setting. *Journal of Automated Reasoning*, 18:85–104, 1997.
4. H. Geuvers, F. Wiedijk, and J. Zwanenburg. Equational Reasoning via Partial Reflection. In *Theorem Proving in Higher Order Logics, 13th International Conference, TPHOLs 2000*, volume 1869 of *LNCS*, pages 162–178, Berlin, Heidelberg, New York, 2000. Springer Verlag.
5. M.J.C. Gordon and T.F. Melham, editors. *Introduction to HOL*. Cambridge University Press, Cambridge, 1993.
6. J.D. Guttman and F.J. Thayer. IMPS: An Interactive Mathematical Proof System. *Journal of Automated Reasoning*, 11:213–248, 1993.
7. John Harrison. Re: Undefined terms.
   Message `<"swan.cl.cam.:266770:950519095422"@cl.cam.ac.uk>` as sent to the QED mailing list,
   `<http://www.ftp.cl.cam.ac.uk/ftp/hvg/qed-project-archive/03xx/0380>`, 1995.
8. Matt Kaufmann, Panagiotis Manolios, and J. Strother Moore. *Computer-Aided Reasoning: An Approach*. Kluwer Academic Publishers, Boston, 2000.
9. J. Kuper. *Partiality in Logic and Computation – Aspects of Undefinedness*. PhD thesis, University of Twente, Dept INF, Enschede, The Netherlands, 1994.
10. M. Muzalewski. *An Outline of PC Mizar*. Fondation Philippe le Hodey, Brussels, 1993. `<http://www.cs.kun.nl/~freek/mizar/mizarmanual.ps.gz>`.
11. R.P. Nederpelt, J.H. Geuvers, and R.C. de Vrijer. *Selected Papers on Automath*, volume 133 of *Studies in Logic and the Foundations of Mathematics*. Elsevier Science, Amsterdam, 1994.
12. T. Nipkow, L.C. Paulson, and M. Wenzel. *Isabelle/HOL – A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, 2002.

13. S. Owre, J. Rushby, and N. Shankar. PVS: A prototype verification system. In D. Kapur, editor, *11th International Conference on Automated Deduction (CADE)*, volume 607 of *LNAI*, pages 748–752, Berlin, Heidelberg, New York, 1992. Springer-Verlag.
14. D.S. Scott. Identity and existence in intuitionistic logic. In M.P. Fourman, C.J. Mulvey, and D.S. Scott, editors, *Applications of Sheaves*, volume 753 of *Lecture Notes in Mathematics*, pages 660–696, Berlin, 1979. Springer-Verlag.
15. Natarajan Shankar and Sam Owre. Principles and Pragmatics of Subtyping in PVS. In Didier Bert, Christine Choppy, and Peter Mosses, editors, *Recent Trends in Algebraic Development Techniques, WADT '99*, volume 1827 of *LNCS*, pages 37–52, Toulouse, France, September 1999. Springer-Verlag.
16. The Coq Development Team. *The Coq Proof Assistant Reference Manual*, 2002. `<ftp://ftp.inria.fr/INRIA/coq/current/doc/Reference-Manual-all.ps.gz>`.
17. A. Troelstra and D. van Dalen. *Constructivism in Mathematics, an Introduction, Vols. 1-2*, volume 121 and 123 of *Studies in Logic and The Foundations of Mathematics*. North-Holland, 1988.
18. L. Wos. *The Automation of Reasoning: An Experimenter's Notebook with Otter Tutorial*. Academic Press, New York, 1996.