Journal of Automated Reasoning manuscript No.
(will be inserted by the editor)

Classical Logic with Partial Functions

Hans de Nivelle

the date of receipt and acceptance should be inserted later

Abstract We introduce a semantics for classical logic with partial functions, in which ill-typed formulas are guaranteed to have no truth value, so that they cannot be used in any form of reasoning. The semantics makes it possible to mix reasoning about types and pre-conditions with reasoning about other properties. This makes it possible to deal with partial functions with preconditions of unlimited complexity. We show that, in spite of its increased complexity, the semantics is still a natural generalization of first-order logic with simple types. If one does not use the increased expressivity, the type system is not stronger than classical logic with simple types.

We will define two sequent calculi for our semantics, and prove that they are sound and complete. The first calculus follows the semantics closely, and hence its completeness proof is fairly straightforward. The second calculus is further away from the semantics, but more suitable for practical use because it has better proof theoretic properties. Its completeness can be shown by proving that proofs from the first calculus can be translated.

Keywords Logic for Partial Functions · Type Systems · Multivalued Semantics

1 Introduction

Partial functions occur frequently in mathematics and in programming, but they are usually ignored in logic. In high school, one is taught that one should not divide by zero. Similarly, one is taught that $\log(0)$ and $\tan\frac{\pi}{2}$ do not exist, and that the inverse of a matrix can be computed only when the matrix is non-singular. When dealing with the semantics of programming languages, partial functions are essential. A pointer can only be dereferenced when it is not the null-pointer. A vector has a first element only if it is non-empty. A file can only be read from if it is in good state and not at end of file. In general, we believe that partiality is more important in programming than in mathematics, because in programming it is not possible to use underspecification. Underspecification means that one assumes that

Hans de Nivelle
Instytut Informatyki Uniwersytetu Wrocławskiego
Wybrzeże Joliot-Curie 15, 50-383, Wrocław, Poland
E-mail: nivelle@ii.uni.wroc.pl

undefined terms have a typeable value, without specifying which value. Using underspecification one would define division as a function from real numbers to real numbers, so that $\frac{1}{0}$ is a real number, but axiomatize it in such a way that nothing can be derived about its concrete value.

The same thing is not possible with programming languages. If one would assume that `*p` has some value, then the program `x = *p; print(x)` would be guaranteed to print something and continue normally. In reality, the program can crash when `*p` has a non-primitive type (for example `std::string`.) Although it is in principle possible to design a programming language without partiality, it would be very unnatural: One would have to attach a default value to every type or to every operator definition. The first solution would be inconsistent with the design philosophy of C^{++}, because the designer of a class should not have to provide a default constructor in the case that no natural default value exists. The second solution would be very tedious, and also conceptually problematic, because in many cases there simply exists no reasonable default value.

In our view, the main purpose of type systems is to avoid meaningless formulas. Because if one accepts that ill-typed formulas can have a truth value (or one is confident to write only well-typed formulas), one does not need any type system at all. In that case, one can use relativization. Unfortunately, if one uses relativization, ill-typed formulas can have unwanted consequences when they are used as axioms. Therefore, we believe it is better to use types. For example, the formula $\forall x,y\colon \mathrm{Real}\ x < y \vee x = y \vee y < x$ can be relativized as $\forall xy\ \mathrm{Real}(x) \wedge \mathrm{Real}(y) \rightarrow x < y \vee x = y \vee y < x$. Both formulas are true in the standard interpretation. The formula $\forall x\colon \mathrm{Real}\ \forall y\colon \mathrm{Complex}\ x < y \vee x = y \vee y < x$ is ill-typed, because $<$ has no meaningful definition on the complex numbers, and a type checker would reject this formula. At the same time, the formula $\forall xy\ \mathrm{Real}(x) \wedge \mathrm{Complex}(y) \rightarrow x < y \vee x = y \vee y < x$ is stronger than the intended meaning. If one tries to prove this formula, then one will probably find out that it is incorrect. If one assumes it as axiom, its incorrectness may go undetected, and result in unwanted logical consequences. In our view, this is the main purpose of type checking: To avoid unwanted consequences of ill-typed axioms.

We want to treat formulas in which preconditions of partial functions are not guaranteed, in the same way as ill-typed formulas, so that such formulas cannot have unwanted consequences when used as axioms. In order to obtain this, we will identify preconditions of partial functions and subtypes. Although there may be philosophical objections, it is often natural to do this. For example, the pointer retrieval operator $*$ can be viewed as a partial function on all pointers, or as a total function on the subtype of non-null pointers. This identification can also be generalized to preconditions of arity > 1. Higher-arity predicates correspond to dependent types. For example, the following declaration $\mathrm{iterator}\colon \Pi \mathrm{x}\colon \mathrm{Type}\ (\mathrm{List\ x}) \rightarrow \mathrm{Type}$, which specifies that every concrete list has a type of its iterators associated to it, can be expressed by the binary predicate $\mathrm{iterator}(l,p)$ denoting that p is an iterator of l.

If one identifies preconditions and subtypes, then underspecification becomes a form of relativation, and it has the same disadvantages that we already mentioned above. (That preconditions cannot be checked in axioms, and that axioms become too strong if one forgets preconditions.)

Because of this we introduced a new approach to classical logic with partial functions in [10], which we call PCL (for *Partial Classical Logic*). It is based on the following considerations: First, there is no natural point, at which one can stop adding complexity to the type system: If one has simple types, then one also wants subtypes, intersection and union types. If one has those, then one also wants partial functions. If one has partial functions, then there is no a priori border on the complexity of preconditions, and there is no simple system in which all possible preconditions can be proven correct. In this way, one naturally arrives at

a system in which unrestricted first-order logic is used for expressing type conditions and preconditions.

Secondly, we want to be able to introduce new functions and predicates in a conservative way. This means that a group of new symbols with some type (and other) properties can only be introduced, after it has been proven that a group of objects with the desired properties exists. In general, one will need additional set axioms or higher-order axioms that allow the introduction of new objects. Assuming that one has a proper set of axioms, one can for example prove that there exist a set $\mathscr{N}$, a $0 \in \mathscr{N}$, and a function **s**, with property $\forall x\ x \in \mathscr{N} \to s(x) \in \mathscr{N}$. These existence proofs can be quite complex and have to carried out in first-order logic (combined with the additional higher-order or set theory principles), It is therefore natural to keep the typing properties in first-order logic, since otherwise, one would need an additional interface to the type system.

The third consideration is that we want to keep the strictness of simple type theory: It should be impossible to assume something that is not well-typed, and derive consequences from it.

Based on the first two considerations, we will use first-order logic for expressing type conditions in PCL. In order to fulfil the strictness condition, we will use the notion of *strong validity*. Intuitively, a sequent $\Gamma_1, \ldots, \Gamma_n \vdash A$ is strongly valid, if it is valid in the usual sense, and in addition each assumption Γ_i is guaranteed to be well-typed if one has assumed the assumptions that occur before it. If one of the assumptions Γ_i among $\Gamma_1, \ldots, \Gamma_n$ is not well-typed, then the sequence $\Gamma_1, \ldots, \Gamma_i$ cannot be completed into a valid sequent, so that the strictness condition is satisfied.

In order to deal with type information in quantifiers, PCL has two additional operators, *lazy implication* $[A]B$ and *lazy conjunction* $\langle A \rangle B$. Their semantics can be obtained by looking at the semantics of quantified formulas in simple type theory, and trying to obtain the same. A formula of form $\forall x{:}X\ P(x)$ can only be assumed when it is well-typed, which means that it must be possible to prove that $P(x)$ is well-typed under the assumption $x{:}X$. Otherwise, strictness would not hold, and it would be possible to derive consequences from ill-typed formulas. Lazy implication has exactly this semantics. The formula $[X(x)]\ P(x)$ is well-typed, only when the assumption $X(x)$ implies that $P(x)$ is well-typed. This means that, before a formula of form $\forall x\ [X(x)]\ P(x)$ can be assumed, one has to show that $P(x)$ is well-typed under the assumption that $X(x)$ is true. Because this check will fail for ill-typed formulas, strictness is ensured. For example, the formula $\forall xy\ [\mathrm{Real}(x) \wedge \mathrm{Complex}(y)]\ x \leq y \vee y \leq x$ will fail type checking, and because of this, never occur in a strongly valid sequent.

Lazy conjunction is similar to lazy implication, but intended for usage with $\exists$. The conjunction $\langle A \rangle B$ is true if both of A and B are true, and well-typed if well-typedness of B can be proven from truth of A. This ensures that $\exists xy\ \langle \mathrm{Real}(x) \wedge \mathrm{Real}(y) \rangle\ x \leq y$ has the same meaning as $\exists x, y{:}\,\mathrm{Real}\ x \leq y$.

In addition to the lazy operators, PCL keeps the standard logical operators $\wedge, \vee, \to$ and $\leftrightarrow$. Their truth value is defined by the standard truth table. Their typing conditions are that $A \star B$ is well-typed if both of A and B are well-typed, independently from each other.

2 Comparison with Existing Approaches

In [3] and [4], the authors introduced a way of dealing with partial functions which they call the *traditional approach to partiality*. (The approach in [13] is very close.) In this approach, **(1)** variables and constants are always defined, **(2)** formulas are always true or false, and **(3)**

atoms (including equalities) containing undefined subterms are always false. The authors claim that this is the approach that is informally used by the majority of mathematicians.

The main difference between the traditional approach and PCL is that in PCL, ill-typed formulas are not propositions, while in the traditional approach, atoms involving undefined terms are false. As a consequence, non-primitive formulas with ill-defined terms can have either truth value, dependent on the polarity of the undefined atoms. Hence, the traditional approach clearly breaks the strictness condition, that it should not be possible to derive anything from an ill-typed axiom.

In addition (and this is a feature which distinguishes PCL from all existing approaches), PCL has no distinction in treatment of type information and treatment of definedness, while the traditional approach has. In the traditional approach, a partial function has a type and a precondition. For example, tan has type $\mathrm{Real} \to \mathrm{Real}$, and its precondition is $\frac{x}{\pi} - \frac{1}{2} \notin \mathrm{Int}$. A term of form $\tan(x)$ is in principle of type Real, but it can be defined or undefined. In PCL, the typing condition for tan is $\forall x\ \mathrm{Real}(x) \wedge \neg\mathrm{Int}(\frac{x}{\pi} - \frac{1}{2}) \to \mathrm{Real}(\tan(x))$, which mixes the type information with the precondition. Instead of being undefined, an ill-typed term of form $\tan(t)$ simply fails to have type Real. If such a term $\tan(t)$ is used in a formula where Real is required, then the formula will eventually fail to produce a Prop. For example, if one has $\forall x,y\ \mathrm{Real}(x) \wedge \mathrm{Real}(y) \to \mathrm{Prop}(x \leq y)$, $\forall x\ \mathrm{Real}(x) \to \mathrm{Real}(\sin(x))$, and $\neg\mathrm{Int}(\frac{t}{\pi} - \frac{1}{2})$ is not provable, then $\sin(\tan(t))$ cannot be proven Real, so that $\sin(\tan(t)) \leq 1$ cannot be proven Prop. So instead of making ill-defined terms undefined, they fail to have a useful type in PCL, which ensures that they cannot occur in strongly valid sequents.

Another commonly used approach, (see for example in [1,7]), is based on three-valued Kleene logic. Three-valued logic is obtained by introducing an extra value **u**, denoting the truth-value for undefined propositions. Truth values are ordered as $\mathbf{f} < \mathbf{u} < \mathbf{t}$. Using this order, $I(P \vee Q)$ can be defined as $\mathrm{MAX}_<(I(P), I(Q))$, $I(\exists x\, P(x))$ as $\mathrm{MAX}(\{I^x_d(P(x)) \mid d \in D\})$, $I(P \wedge Q)$ can be defined as $\mathrm{MIN}_<(I(P), I(Q))$, and $I(\forall x\, P(x))$ is defined as $\mathrm{MIN}(\{I^x_d(P(x)) \mid d \in D\})$. Negation can be defined from $[\mathbf{f} \Rightarrow \mathbf{t},\ \mathbf{u} \Rightarrow \mathbf{u},\ \mathbf{t} \Rightarrow \mathbf{f}]$. The remaining operators $\to$ and $\leftrightarrow$ can be introduced through standard equivalences.

The well-definedness approach of [2,8] is closely related, although at first it may appear different. In the well-definedness approach one has to prove that a formula is well-defined before it can be used. However, if one inspects the translation from three-valued logic to two-valued logic in [1], the result of the translation is the logic of [8], so that the well-definendness approach is equivalent to three-valued logic.

Three-valued logic (and the well-definedness approach) are closer to our aim of strictness, but there is still a fundamental philosophical difference: The three-valued approach treats ill-defined formulas as formulas with unknown truth-value, where PCL treats them as error values. The justification for defining $(\mathbf{t} \vee \mathbf{u}) = \mathbf{t}$ is the fact that whichever value **u** will eventually take, the presence of **t** ensures that the disjunction will be true. In a completely strict setting, nothing should be assumed about formulas containing ill-typed formulas, because they are errors, not unknowns.

The difference in semantics between PCL and the three-valued approach can be observed in sequents where some of the premises are ill-typed, but not required for validity of the sequent. (Alternatively, one could say that three-valued Kleene logic allows weakening by ill-typed formulas, while PCL does not.) In Kleene logic, validity of the sequent $\Gamma_1, \ldots, \Gamma_n \vdash A$ means that $I(\neg\Gamma_1 \vee \cdots \vee \neg\Gamma_n \vee A) = \mathbf{t}$ in every interpretation I. It is easily seen that validity does not depend on the order of the Γ_i, and that validity is preserved, when formulas are added to Γ at random positions, even when they are ill-typed. This does not hold in PCL. If one adds an ill-typed formula F between Γ_i and Γ_{i+1}, then the sequent is not strongly valid anymore, because the sequent $\Gamma_1, \ldots, \Gamma_i \vdash \mathrm{Prop}(F)$ is not valid.

We conclude that three-valued Kleene semantics has an intermediate form of strictness: It is possible to make derivations involving ill-typed formulas, but not possible to use the ill-typed formula in the derivation. On the other hand, PCL has the same level of strictness as simple type systems. To illustrate the difference, consider the ill-typed sequent $0\colon \mathrm{Nat},\ s\colon \mathrm{Nat} \to \mathrm{Nat},\ \leq\colon \mathrm{Nat} \to \mathrm{Nat} \to \mathrm{Prop},\ \mathrm{empty}\colon \mathrm{String} \to \mathrm{Prop},\ \mathrm{empty}(0) \vdash s(0) = s(0)$. A standard type system would not accept it. The corresponding formula $\neg\mathrm{empty}(0) \vee s(0) = s(0)$ is valid in three-valued semantics, despite the fact that $I(\mathrm{empty}(0))$ may be **u**. (We assumed that the theory has no a priori type system, so that ill-typedness is handled by **u**. Otherwise, the example would have to involve preconditions of partial functions, which would make it more complicated, and it would be harder to compare with standard type systems.) At the same time, the sequent

$$\mathrm{Nat}(0),\ \forall x\ \mathrm{Nat}(x) \to \mathrm{Nat}(s(x)),\ \forall xy\ \mathrm{Nat}(x) \wedge \mathrm{Nat}(y) \to \mathrm{Prop}(x \leq y),\\ \forall xy\ \mathrm{String}(x) \to \mathrm{Prop}(\mathrm{empty}(x)),\ \mathrm{empty}(0) \vdash s(0) = s(0)$$

is not strongly valid in PCL. We conclude that PCL carries over the strictness of simple type theory to preconditions of partial functions, while three valued semantics is more liberal.

3 Syntax and Semantics of PCL

We first introduce the syntax of PCL, and after that its semantics. Since type conditions and preconditions are expressed by formulas inside the logic itself, there is no need to consider types in the definition of formulas. There is also no need to distinguish atoms and terms at the syntactic level, because this distinction appears later when certain terms are declared to have Prop.

Definition 1 The set of *terms* of PCL (*Partial Classical Logic*) is recursively defined by the following rule: If $t_1, \ldots, t_n$ are terms (with $n \geq 0$), and f is a function symbol with arity n, then $f(t_1, \ldots, t_n)$ is also a term. We call function symbols with arity 0 *constants* or *variables*, dependent on how they are used.

Using the set of terms, we define the set of formulas of PCL recursively as follows:

- $\bot$ and $\top$ are formulas.
- Every term A is a formula. We will call formulas of this form *atoms*.
- If t_1, t_2 are terms, then $t_1 = t_2$ is a formula.
- If A is a formula, then $\neg A$ is a formula.
- If A and B are formulas, then $A \wedge B$, $A \vee B$, $A \to B$, and $A \leftrightarrow B$ are formulas.
- If A and B are formulas, then $[A]B$ and $\langle A\rangle B$ are formulas.
- If x is a variable, A is a formula, then $\forall x\ A$ and $\exists x\ A$ are formulas.
- If A is a formula, then $\mathrm{Prop}(A)$ is a formula.

The intuitive meaning of $\mathrm{Prop}(F)$ is 'F has a truth-value'.

We introduce the semantics of PCL. As with the syntax, interpretations have no built-in typing. An interpretation has a single domain D, which contains all relevant domains of discourse. The domain has two dedicated truth values **f** and **t**, but no dedicated value for undefined terms or undefined predicates. Instead, an ill-typed term produces a result that is not included in any type, and which does not satisfy any precondition. When an ill-typed term is used in a predicate, its result will lie outside of the domain of the predicate. As a

consequence, the predicate will not produce a Prop so that the result cannot be used as a formula. This is the way in which undefinedness and partiality are handled in PCL.

Before we continue, we introduce a lattice on non-truth values. The semantics could be introduced without it, but the lattice makes it possible to formulate meta properties as equivalences. Without it, the behaviour of the logical operators on non-truth values would be completely unpredictable. Then, for example $I(A) = I(A \wedge A)$ would not hold as equality. This would have no effect on the provable formulas, because the equality would still hold for the truth values, but it is more elegant to have the equality for the complete domain. The lattice plays no role in reasoning, and we do not intend to use it for abstraction, as proposed in [6].

Definition 2 Let S be a set. A relation $\sqsubseteq$ is called *a partial order* if it meets the following requirements: **(1)** For all $s \in S$, $s \sqsubseteq s$. **(2)** For all s_1, s_2, s_3, $s_1 \sqsubseteq s_2 \wedge s_2 \sqsubseteq s_3 \Rightarrow s_1 \sqsubseteq s_3$. **(3)** For all s_1, s_2, $s_1 \sqsubseteq s_2 \wedge s_2 \sqsubseteq s_1 \Rightarrow s_1 = s_2$. Let S' be a subset of S. We call $s \in S$ a *lower bound* of S' if for all $s' \in S'$, $s \sqsubseteq s'$. We call s a greatest lower bound of S' if s is a lower bound of S', and for every lower bound $\hat{s}$ of S', we have $\hat{s} \sqsubseteq s$.
We write $\sqcap S'$ for the greatest lower bound of S', if it exists. If S' is finite, we write $s_1 \sqcap s_2 \sqcap \cdots \sqcap s_n$ instead of $\sqcap\{s_1, s_2, \ldots, s_n\}$.

It is easily checked that the greatest lower bound is unique if it exists.

Definition 3 An interpretation $I = (D, \mathbf{f}, \mathbf{t}, \sqsubseteq, [\,])$ is defined by

- A domain D.
- Two distinct truth constants $\mathbf{f}$ and $\mathbf{t}$, such that both of $\mathbf{f}, \mathbf{t} \in D$,
- A partial order $\sqsubseteq$ on $D \backslash \{\mathbf{f}, \mathbf{t}\}$, s.t. every non-empty $D' \subseteq D \backslash \{\mathbf{f}, \mathbf{t}\}$ has a greatest lower bound $\sqcap D'$, which is in $D \backslash \{\mathbf{f}, \mathbf{t}\}$.
- a function $[\,]$ that interprets function symbols as follows: If f is a function symbol with arity n, then $[f]$ is a total function from D^n to D.

As said before, the only role of the partial order $\sqsubseteq$ is to obtain predictable behaviour of the logical operators when they are applied on non-Boolean objects.

Definition 4 Let $I = (D, \mathbf{f}, \mathbf{t}, \sqsubseteq, [\,])$ be an interpretation. We recursively define the *interpretation* $I(F)$ of a formula F as follows:

- $I(\bot) = \mathbf{f}$, $I(\top) = \mathbf{t}$.
- For a formula of form $f(t_1, \ldots, t_n)$, $I(\ f(t_1, \ldots, t_n)\ ) = [f](I(t_1), \ldots, I(t_n))$.
- If $I(t_1) = I(t_2)$, then $I(t_1 = t_2) = \mathbf{t}$. Otherwise, $I(t_1 = t_2) = \mathbf{f}$.
- If $I(A) = \mathbf{t}$, then $I(\neg A) = \mathbf{f}$. If $I(A) = \mathbf{f}$, then $I(\neg A) = \mathbf{t}$. Otherwise $I(\neg A) = I(A)$.
- We characterize the strict binary operators:
  - If $I(A) \in \{\mathbf{f}, \mathbf{t}\}$, and $I(B) \notin \{\mathbf{f}, \mathbf{t}\}$, then $I(A \wedge B) = I(A \vee B) = I(A \to B) = I(A \leftrightarrow B) = I(B)$.
  - If $I(A) \notin \{\mathbf{f}, \mathbf{t}\}$, and $I(B) \in \{\mathbf{f}, \mathbf{t}\}$, then $I(A \wedge B) = I(A \vee B) = I(A \to B) = I(A \leftrightarrow B) = I(A)$.
  - If both $I(A), I(B) \notin \{\mathbf{f}, \mathbf{t}\}$, then $I(A \wedge B) = I(A \vee B) = I(A \to B) = I(A \leftrightarrow B) = I(A) \sqcap I(B)$.
  - If both of $I(A), I(B) \in \{\mathbf{f}, \mathbf{t}\}$, then $\wedge, \vee, \to, \leftrightarrow$ are characterized by the following (standard) truth table:

$I(A)$	$I(B)$	$I(A \wedge B)$	$I(A \vee B)$	$I(A \rightarrow B)$	$I(A \leftrightarrow B)$
f	**f**	**f**	**f**	**t**	**t**
f	**t**	**f**	**t**	**t**	**f**
t	**f**	**f**	**t**	**f**	**f**
t	**t**	**t**	**t**	**t**	**t**

- We characterize the lazy binary operators:
  - If $I(A) = \mathbf{f}$, then $I(\ [A]B\ ) = \mathbf{t}$, and $I(\ \langle A\rangle B\ ) = \mathbf{f}$.
  - If $I(A) \notin \{\mathbf{f},\mathbf{t}\}$, and $I(B) \in \{\mathbf{f},\mathbf{t}\}$, then $I(\ [A]B\ ) = I(\ \langle A\rangle B\ ) = I(A)$.
  - If both $I(A), I(B) \notin \{\mathbf{f},\mathbf{t}\}$, then $I(\ [A]B\ ) = I(\ \langle A\rangle B\ ) = I(A) \sqcap I(B)$.
  - If $I(A) = \mathbf{t}$, then $I(\ [A]B\ ) = I(\ \langle A\rangle B\ ) = I(B)$.
- Next come the quantifiers: Let x be some variable. Let F be a formula. Let $R = \{\ I^x_d(F) \mid d \in D\}$, where I^x_d is obtained as usual by changing the value of x in $[\ ]$ to d.
  - If $R \not\subseteq \{\mathbf{f},\mathbf{t}\}$, then $I(\ \forall x\ F\ ) = I(\ \exists x\ F\ ) = \sqcap(R \backslash \{\mathbf{f},\mathbf{t}\})$.
  - If $R = \{\mathbf{f}\}$, then $I(\ \forall x\ F\ ) = I(\ \exists x\ F\ ) = \mathbf{f}$.
  - If $R = \{\mathbf{t}\}$, then $I(\ \forall x\ F\ ) = I(\ \exists x\ F\ ) = \mathbf{t}$.
  - If $R = \{\mathbf{f},\mathbf{t}\}$, then $I(\ \forall x\ F\ ) = \mathbf{f}$ and $I(\ \exists x\ F\ ) = \mathbf{t}$.
- It remains to characterize Prop. If $I(A) \in \{\mathbf{f},\mathbf{t}\}$, then $I(\ \mathrm{Prop}(A)\ ) = \mathbf{t}$. Otherwise, $I(\ \mathrm{Prop}(A)\ ) = \mathbf{f}$.

Definition 5 A *context* is a finite sequence of formulas $\Gamma_1, \ldots, \Gamma_n$. A *sequent* is an object of form $\Gamma \vdash A$, in which Γ is a context and A is a formula.

We now precisely define the notion of strong validity that was mentioned in the introduction. There we said that a sequent is strongly valid iff it is valid in the usual sense, and every of its premises is necessarily Prop when all of the premises before it are true. This notion can be equivalently formulated as follows:

Definition 6 Let $\Gamma_1, \ldots, \Gamma_n \vdash A$ be a sequent. We call $\Gamma_1, \ldots, \Gamma_n \vdash A$ *valid* if in every interpretation $I = (D, \mathbf{f}, \mathbf{t}, \sqsubseteq, [\ ])$, s.t. $I(\Gamma_1) = \cdots = I(\Gamma_n) = \mathbf{t}$, we also have $I(A) = \mathbf{t}$. We call the sequent $\Gamma_1, \ldots, \Gamma_n \vdash A$ *strongly valid*, if it is valid, and in addition the context $\Gamma_1, \ldots, \Gamma_n$ has the following property: In every interpretation I, we have either $I(\Gamma_i) = \mathbf{t}$ for all i with $1 \leq i \leq n$, or we have $I(\Gamma_i) = \mathbf{f}$ for the first i with $1 \leq i \leq n$ that has $I(\Gamma_i) \neq \mathbf{t}$.

We prove that strong validity as defined in Definition 6, is equivalent to the informal notion that we used in the introduction. This ensures that strong validity ensures strictness, which is one of the design goals of PCL.

Theorem 1 *Let $\Gamma_1, \ldots, \Gamma_n \vdash A$ be a sequent. Then $\Gamma_1, \ldots, \Gamma_n \vdash A$ is strongly valid iff $\Gamma_1, \ldots, \Gamma_n \vdash A$ is valid and for every i with $1 \leq i \leq n$, the sequent $\Gamma_1, \ldots, \Gamma_{i-1} \vdash \mathrm{Prop}(\Gamma_i)$ is valid.*

Proof Assume that $\Gamma_1, \ldots, \Gamma_n \vdash A$ is strongly valid. It is clear that this implies that the sequent is valid. In order to show that each $\Gamma_1, \ldots, \Gamma_{i-1} \vdash \mathrm{Prop}(\Gamma_i)$ is valid, we assume that in some interpretation I, we have $I(\Gamma_1) = \cdots = I(\Gamma_{i-1}) = \mathbf{t}$. Using this assumption, we have to show that $I(\mathrm{Prop}(\Gamma_i)) = \mathbf{t}$. If $I(\Gamma_i) = \mathbf{t}$, then we are done. Otherwise, by strong validity of $\Gamma_1, \ldots, \Gamma_n \vdash A$, we have $I(\Gamma_i) = \mathbf{f}$, so that the proof is complete.

Now assume that $\Gamma_1, \ldots, \Gamma_n \vdash A$ is valid, and for every i with $1 \leq i \leq n$, the sequent $\Gamma_1, \ldots, \Gamma_{i-1} \vdash \mathrm{Prop}(\Gamma_i)$ is valid. We have to show that $\Gamma_1, \ldots, \Gamma_n \vdash A$ is strongly valid. Let I be an arbitrary interpretation. If $I(\Gamma_1) = \cdots = I(\Gamma_n) = \mathbf{t}$, then we are done. Otherwise, let Γ_i be the first formula in the context $\Gamma_1, \ldots, \Gamma_n$ for which $I(\Gamma_i) \neq \mathbf{t}$. Because $\Gamma_1, \ldots, \Gamma_{i-1} \vdash \mathrm{Prop}(\Gamma_i)$ is valid, it must be the case that $I(\Gamma_i) = \mathbf{f}$. This completes the proof.

The sequent $A \vdash A$ is valid, but not strongly valid, because there exists an interpretation I with $I(A) = \mathbf{e}$, and $\mathbf{e} \notin \{\mathbf{f}, \mathbf{t}\}$. Alternatively, one can observe that the sequent $\vdash \mathrm{Prop}(A)$ fails in I. The sequent $\mathrm{Prop}(A), A \vdash A$ is strongly valid because it is valid, and if $I(A) \notin \{\mathbf{f}, \mathbf{t}\}$, then $I(\mathrm{Prop}(A)) = \mathbf{f}$. Similarly, the sequent $\vdash A \vee \neg A$ is valid, but not strongly valid. The sequent $\mathrm{Prop}(A) \vdash A \vee \neg A$ is strongly valid.

Theorem 1 shows that PCL uses type information in contexts in a strict way. Next we prove that usage of type information in quantifiers is connected to usage of type information in contexts in the same way as in standard type systems. In a standard type system, a quantified formula of form $\forall x{:}\, T\ P(x)$ is well-typed in a context Γ iff $\Gamma, x{:}\, T \vdash P(X){:}\,\mathrm{Prop}$ is provable. We show that PCL has the same property, as long as $[\,]$ or $\langle\,\rangle$ is used for introducing the type information.

Theorem 2 *Let Γ be a context, let x be a variable that is not free in Γ. The following three statements are equivalent:*

1. $\Gamma \vdash \mathrm{Prop}(\ \forall x\ [A(x)]B(x)\ )$ *is valid,*
2. $\Gamma \vdash \mathrm{Prop}(\ \exists x\ \langle A(x)\rangle B(x)\ )$ *is valid,*
3. $\Gamma \vdash \mathrm{Prop}(A(x))$ *and* $\Gamma,\ A(x) \vdash \mathrm{Prop}(B(x))$ *are valid.*

Proof Equivalence of **(1)** and **(2)** can be proven by showing that $I(\mathrm{Prop}(\forall x\ [A(x)]B(x))) = I(\mathrm{Prop}(\exists x\ \langle A(x)\rangle B(x)))$, which in turn can be proven by checking Definition 4.

Assume that **(1)** holds, and let $I = (D, \mathbf{f}, \mathbf{t}, \sqsubseteq, [\,])$ be an arbitrary interpretation in which $I(\Gamma) = \mathbf{t}$. We first show that $I(\mathrm{Prop}(A(x))) = \mathbf{t}$. Since we know that $I(\mathrm{Prop}(\forall x\ [A(x)]B(x))) = \mathbf{t}$, we know that $I(\forall x\ [A(x)]B(x)) \in \{\mathbf{f}, \mathbf{t}\}$. From Definition 4 follows that for every $d \in D$, we have $I^x_d([A(x)]B(x)) \in \{\mathbf{f}, \mathbf{t}\}$. If we select $d = [x]$, we have $I^x_{[x]}([A(x)]B(x)) \in \{\mathbf{f}, \mathbf{t}\}$. Since $I^x_{[x]} = I$, we have $I([A(x)]B(x)) \in \{\mathbf{f}, \mathbf{t}\}$. It can be checked from Definition 4 that $I(A(x)) \in \{\mathbf{f}, \mathbf{t}\}$, which implies that $I(\mathrm{Prop}(A(x))) = \mathbf{t}$.

For the second part of **(3)**, assume that I is an arbitrary interpretation in which $I(\Gamma) = I(A(x)) = \mathbf{t}$. By the same reasoning as before, we obtain that $I([A(x)]B(x)) \in \{\mathbf{f}, \mathbf{t}\}$. It follows from Definition 4 that $I(B(x)) \in \{\mathbf{f}, \mathbf{t}\}$, so that we have shown **(3)**.

Next we assume **(3)**, so we assume that $\Gamma \vdash \mathrm{Prop}(A(x))$ and $\Gamma,\ A(x) \vdash \mathrm{Prop}(B(x))$ are valid sequents. Assume that for some interpretation $I = (D, \mathbf{f}, \mathbf{t}, \sqsubseteq, [\,])$, we have $I(\Gamma) = \mathbf{t}$. We have to show that $I(\mathrm{Prop}(\forall x\ [A(x)]B(x)) = \mathbf{t}$, which can be done by showing that for every $d \in D$, $I^x_d([A(x)]B(x)) \in \{\mathbf{f}, \mathbf{t}\}$. So let $d \in D$. Since $I(\Gamma) = I^x_d(\Gamma)$, we have $I^x_d(\mathrm{Prop}(A(x)) = \mathbf{t}$, which implies that either $I^x_d(A(x)) = \mathbf{f}$, or $I^x_d(A(x)) = \mathbf{t}$. If $I^x_d(A(x)) = \mathbf{f}$, we have $I^x_d([A(x)]B(x)) = \mathbf{t}$, so that we are done. Otherwise, we have $I^x_d(B(x)) \in \{\mathbf{f}, \mathbf{t}\}$, so that we are also done.

Theorems 1 and 2 together show that PCL uses type information (and preconditions of partial functions) in the same way as standard type systems. It is actually possible to prove stronger properties, for example that if one translates a simple type system into first-order logic, then PCL will accept exactly the same type judgements as simple type theory. The same is provable for type systems with intersection and union types. We omit these results here, because of lack of space.

4 Sequent Calculus for PCL

We want to define a sequent calculus that is able to model strong validity. This is easier if one uses a one-sided calculus, in which sequents are refuted, instead of proven. The reasons for

this are the following: Validity of $\mathrm{Prop}(A) \vdash A \vee \neg A$ and the fact that the semantics is based on truth-values, suggest that PCL is essentially classical (in contrast to intuitionistic). At the same time, the notion of strong validity depends on the order of formulas in the sequent. Allowing formulas to freely move from the premise to the conclusion in a sequent, which would be needed for classical $\neg$-rules, and simultaneously keeping track of the order of the formulas in the sequent, is tedious. It can be avoided by using one-sided sequents.

Definition 7 A *one-sided sequent* is an object of form $\Gamma_1, \ldots, \Gamma_n \vdash$, in which $\Gamma_1, \ldots, \Gamma_n$ ($n \geq 0$) is a sequence of formulas. We say that $\Gamma_1, \ldots, \Gamma_n \vdash$ *fails* in an interpretation I if there is an i, $(1 \leq i \leq n)$, s.t. $I(\Gamma_i) \neq \mathbf{t}$. We will usually write 'sequent' instead of 'one-sided sequent', since it is always clear from the form which type is meant.

We say that $\Gamma \vdash$ *fails strongly* in I if there is an i, $(1 \leq i \leq n)$, s.t. $I(\Gamma_i) = \mathbf{f}$ and for all j, $(1 \leq j < i)$, $I(\Gamma_j) = \mathbf{t}$. If we want to stress that Γ_i is the first formula in Γ with $I(\Gamma_i) \neq \mathbf{t}$ (which implies that $I(\Gamma_i) = \mathbf{f}$), then we say that Γ *fails strongly at* Γ_i in I.

We call the one-sided sequent $\Gamma \vdash$ *unsatisfiable* if it fails in every interpretation. We call Γ *strongly unsatisfiable* if it fails strongly in every interpretation.

Theorem 3 *Let $\Gamma \vdash A$ be a sequent. $\Gamma \vdash A$ is strongly valid if and only if the one-sided sequent $\Gamma, \neg A \vdash$ is strongly unsatisfiable.*

Proof Write $\Gamma = \Gamma_1, \ldots, \Gamma_n \vdash$ with $n \geq 0$. For convenience, define $\Gamma_{n+1} := \neg A$.

Assume that $\Gamma \vdash A$ is strongly valid. We have to show that the sequent $\Gamma_1, \Gamma_2, \ldots, \Gamma_{n+1} \vdash$ is strongly unsatisfiable. Let I be an arbitrary interpretation. We have to show that there exists an i, $(1 \leq i \leq n+1)$ with property $\Phi(i)$, where $\Phi(i)$ is the property that $I(\Gamma_i) = \mathbf{f}$, and for all j, $(1 \leq j < i)$, $I(\Gamma_j) = \mathbf{t}$. We distinguish two cases:

- If for all i, $1 \leq i \leq n$, $I(\Gamma_i) = \mathbf{t}$, then it follows from validity of $\Gamma \vdash A$ that $I(A) = \mathbf{t}$, so that $I(\neg A) = \mathbf{f}$. Since $\neg A = \Gamma_{n+1}$, we have $\Phi(n+1)$.
- If there is an i with $1 \leq i \leq n$, s.t. $I(\Gamma_i) \neq \mathbf{t}$, then by strong validity of $\Gamma \vdash A$ (see Definition 6), we have $I(\Gamma_i) = \mathbf{f}$ for the first i with $I(\Gamma_i) \neq \mathbf{t}$. It follows that we have $\Phi(i)$.

In order to show the other direction, assume that $\Gamma_1, \ldots, \Gamma_n, \Gamma_{n+1} \vdash$ is strongly unsatisfiable. We first show that $\Gamma_1, \ldots, \Gamma_n \vdash A$ is valid. Let I be an interpretation. Assume that $I(\Gamma_1) = I(\Gamma_2) = \cdots = I(\Gamma_n) = \mathbf{t}$. It follows from the strong unsatisfiability of $\Gamma_1, \ldots, \Gamma_n, \Gamma_{n+1} \vdash$ that $I(\Gamma_{n+1}) = \mathbf{f}$, so that $I(A) = \mathbf{t}$. Next we show the additional property that makes $\Gamma_1, \ldots, \Gamma_n \vdash A$ strongly valid. If no i has $I(\Gamma_i) \neq \mathbf{t}$, then we are done. Otherwise, let i be the first position where $I(\Gamma_i) \neq \mathbf{t}$. If we would have $I(\Gamma_i) \neq \mathbf{f}$, then this would contradict strong unsatisfiability of the sequent $\Gamma_1, \ldots, \Gamma_n, \Gamma_{n+1} \vdash$, so that the proof is complete.

Using Theorem 3, the conclusion of a sequent can be moved to the left hand side, after which it can be treated in the same way as the other premises. This has the advantage that one can delete half of the rules from the sequent calculus, and it avoids the burden of keeping track of the order of formulas spread between the premises and the conclusions. In order to further simplify the calculus, we use the reduction rules in Figure 1 and Figure 2. Figure 1 contains rules for pushing negation inwards, while Figure 2 contains rules for pushing Prop inwards. Most rules in Figure 1 look familiar, but their validity still needs to be checked in the context of PCL. It can be checked (by case analysis) that for every interpretation I, for each rule $A \Rightarrow B$ in Figure 1 or Figure 2, we have $I(A) = I(B)$, so that the equivalences can be freely used in proofs. Figure 1 and Figure 2 ensure that $\neg$ or Prop never needs to be the main operator of a formula. The only cases where Prop and $\neg$ cannot be eliminated are

Fig. 1 Reduction Rules for $\neg$

$$
\begin{array}{lcllcl}
\neg\bot & \Rightarrow & \top & \neg\top & \Rightarrow & \bot \\
\neg\neg A & \Rightarrow & A & & & \\
\neg(\ \langle A\rangle B\ ) & \Rightarrow & [A]\neg B & \neg(\ [A]B\ ) & \Rightarrow & \langle A\rangle\neg B \\
\neg(A\wedge B) & \Rightarrow & \neg A\vee\neg B & \neg(A\vee B) & \Rightarrow & \neg A\wedge\neg B \\
\neg(A\rightarrow B) & \Rightarrow & A\wedge\neg B & \neg(A\leftrightarrow B) & \Rightarrow & A\leftrightarrow\neg B \\
\neg\forall x\,F, & \Rightarrow & \exists x\,\neg F & \neg\exists x\,F & \Rightarrow & \forall x\,\neg F
\end{array}
$$

Fig. 2 Reduction Rules for Prop

$$
\begin{array}{lcll}
\mathrm{Prop}(\top) & \Rightarrow & \top & \\
\mathrm{Prop}(\bot) & \Rightarrow & \top & \\
\mathrm{Prop}(\neg A) & \Rightarrow & \mathrm{Prop}(A) & \\
\mathrm{Prop}(\mathrm{Prop}(A)) & \Rightarrow & \top & \\
 & & & \\
\mathrm{Prop}(A\wedge B) & \Rightarrow & \mathrm{Prop}(A)\wedge\mathrm{Prop}(B) & (\text{or }\langle\ \mathrm{Prop}(A)\ \rangle\ \mathrm{Prop}(B)\ ) \\
\mathrm{Prop}(A\vee B) & \Rightarrow & \mathrm{Prop}(A)\wedge\mathrm{Prop}(B) & (\text{or }\langle\ \mathrm{Prop}(A)\ \rangle\ \mathrm{Prop}(B)\ ) \\
\mathrm{Prop}(A\rightarrow B) & \Rightarrow & \mathrm{Prop}(A)\wedge\mathrm{Prop}(B) & (\text{or }\langle\ \mathrm{Prop}(A)\ \rangle\ \mathrm{Prop}(B)\ ) \\
\mathrm{Prop}(A\leftrightarrow B) & \Rightarrow & \mathrm{Prop}(A)\wedge\mathrm{Prop}(B) & (\text{or }\langle\ \mathrm{Prop}(A)\ \rangle\ \mathrm{Prop}(B)\ ) \\
 & & & \\
\mathrm{Prop}(\ \langle A\rangle B\ ) & \Rightarrow & \langle\ \mathrm{Prop}(A)\ \rangle(\ A\rightarrow\mathrm{Prop}(B)\ ) & \\
\mathrm{Prop}(\ [A]B\ ) & \Rightarrow & \langle\ \mathrm{Prop}(A)\ \rangle(\ A\rightarrow\mathrm{Prop}(B)\ ) & \\
 & & & \\
\mathrm{Prop}(\forall x\,F) & \Rightarrow & \forall x\ \mathrm{Prop}(F) & \\
\mathrm{Prop}(\exists x\,F) & \Rightarrow & \forall x\ \mathrm{Prop}(F) & \\
 & & & \\
\mathrm{Prop}(t_1=t_2) & \Rightarrow & \top &
\end{array}
$$

in formulas of form $\phi_1(\phi_2(A))$, where A is an atom, ϕ_2 is either Prop or nothing, and ϕ_1 is either $\neg$ or nothing. Such formulas play the same role as literals in first-order logic. One can either simplify the sequent completely before proof search, or apply the rules 'lazily,' i.e. only when $\neg$ or Prop stands in the way of a rule application.

Figure 3 contains the rules of the sequent calculus Seq_{PCL}. Most rules look familiar, but there are pitfalls. For example, the rule for $\wedge$-introduction would be unsound if the second premise would be removed. It would then be possible that in some interpretation $I=(D,\mathbf{f},\mathbf{t},\sqsubseteq,[\,])$, one has $I(\Gamma_1)=\mathbf{t}$, $I(A)=\mathbf{f}$, and $I(B)\notin\{\mathbf{f},\mathbf{t}\}$. In that case the left premise would fail strongly, while the conclusion would fail only weakly. Similarly, the rule for $\forall$-introduction would be unsound if one would not keep a copy of $\forall x\,P(x)$ in the premise before $P(t)$. It could happen that in some interpretation I, $I(P(t))=\mathbf{t}$, while at the same time $I(\forall x\,P(x))\notin\{\mathbf{f},\mathbf{t}\}$.

If one would remove the A from the second premise in $[\,]$-introduction, the rule would still be sound, but become too weak for completeness. The problem would show up when $\mathrm{Prop}(B)$ depends on A.

In contrast to standard first-order logic, a sequent of form $\Gamma,A,\neg A\vdash$ is not automatically an axiom. It is possible that $\Gamma\vdash$ fails weakly, or $I(\Gamma)=\mathbf{t}$ and $I(A)\notin\{\mathbf{f},\mathbf{t}\}$. Both cases are covered by requiring the additional sequent $\Gamma,\ \neg\mathrm{Prop}(A)\vdash$.

We will prove soundness of the rules for $\langle\ \rangle$, $\vee$, and $\exists$. Most of the other rules can be reduced to $\langle\ \rangle$ and $\vee$, by using the equivalences in Figure 4. The remaining rules can be checked by case analysis.

Theorem 4 *Let $\Gamma_{\langle A\rangle B}$ be a sequent of form $\Gamma_1,\ldots,\Gamma_m,\ \langle A\rangle B,\ \Gamma'_1,\ldots,\Gamma'_n\vdash$. Let $\Gamma_{A,B}$ be the sequent $\Gamma_1,\ldots,\Gamma_m,A,B,\Gamma'_1,\ldots,\Gamma'_n\vdash$. Let I be an interpretation. Then $\Gamma_{\langle A\rangle B}$ fails strongly in I iff $\Gamma_{A,B}$ fails strongly in I.*

Fig. 3 Rules of Seq_{PCL} :

Rules for $\langle\ \rangle$ and $\vee$

$$\frac{\Gamma_1, A, B, \Gamma_2 \vdash}{\Gamma_1,\ \langle A\rangle\, B,\ \Gamma_2 \vdash} \qquad\qquad \frac{\Gamma_1, A, \Gamma_2 \vdash \qquad \Gamma_1, B, \Gamma_2 \vdash}{\Gamma_1,\ A \vee B,\ \Gamma_2 \vdash}$$

Rules for $\wedge$ and $[\,]$

$$\frac{\Gamma_1, A, B, \Gamma_2 \vdash \qquad \Gamma_1, B, A, \Gamma_2 \vdash}{\Gamma_1,\ A \wedge B,\ \Gamma_2 \vdash} \qquad\qquad \frac{\Gamma_1,\ \neg A,\ \Gamma_2 \vdash \qquad \Gamma_1, A, B, \Gamma_2 \vdash}{\Gamma_1,\ [A]B,\ \Gamma_2 \vdash}$$

Rules for $\rightarrow$ and $\leftrightarrow$

$$\frac{\Gamma_1,\ \neg A,\ \Gamma_2 \qquad \Gamma_1, B, \Gamma_2 \vdash}{\Gamma_1,\ A \rightarrow B,\ \Gamma_2 \vdash} \qquad\qquad \frac{\Gamma_1,\ A, B,\ \Gamma_2 \vdash \qquad \Gamma_1,\ \neg B, \neg A,\ \Gamma_2 \vdash}{\Gamma_1,\ A \leftrightarrow B,\ \Gamma_2 \vdash}$$

Rules for $\forall$ and $\exists$

$$\frac{\Gamma_1,\ \forall x\, P(x),\ P(t),\ \Gamma_2 \vdash}{\Gamma_1,\ \forall x\, P(x),\ \Gamma_2 \vdash} \qquad\qquad \frac{\Gamma_1,\ P(y),\ \Gamma_2 \vdash}{\Gamma_1,\ \exists x\, P(x),\ \Gamma_2 \vdash}$$

(In the $\forall$-rule, t must be a term. In the $\exists$-rule, y must be not free in Γ_1 or Γ_2.)

Equivalence If $A \Rightarrow B$ is an instance of one of the rules in Figure 2 or Figure 1, then the following derivation is possible:

$$\frac{\Gamma_1, B, \Gamma_2 \vdash}{\Gamma_1, A, \Gamma_2 \vdash}$$

Axioms

$$\frac{\Gamma,\ \neg\mathrm{Prop}(A) \vdash}{\Gamma, A, \neg A \vdash} \qquad\qquad \frac{\Gamma,\ \neg\mathrm{Prop}(A) \vdash}{\Gamma, \neg A, A \vdash} \qquad\qquad \frac{}{\bot \vdash}$$

Weakening

$$\frac{\Gamma_1, \neg\mathrm{Prop}(A) \vdash \qquad \Gamma_1, \Gamma_2 \vdash}{\Gamma_1, A, \Gamma_2 \vdash} \qquad\qquad \frac{\Gamma \vdash}{\Gamma, A \vdash} \qquad\qquad \frac{\Gamma_1, \Gamma_2 \vdash}{\Gamma_1, \top, \Gamma_2 \vdash}$$

(In the first two rules, A can be positive or negative.)

Contraction, Cut

$$\frac{\Gamma_1, A, \Gamma_2, A, \Gamma_3 \vdash}{\Gamma_1, A, \Gamma_2, \Gamma_3 \vdash} \qquad\qquad \frac{\Gamma_1,\ \neg A,\ \Gamma_2 \vdash \qquad \Gamma_1, A, \Gamma_2 \vdash}{\Gamma_1, \Gamma_2 \vdash}$$

(A can be positive or negative.)

Equality

$$\frac{\Gamma_1,\ t_1 \neq t_2 \vdash \qquad \Gamma_1, \Gamma_2[t_1] \vdash}{\Gamma_1, \Gamma_2[t_2] \vdash} \qquad\qquad \frac{}{t \neq t \vdash}$$

Fig. 4 Reduction of remaining rules to $\langle\ \rangle$ and $\vee$

$\wedge$	$A \wedge B$	$\Rightarrow$	$(\ \langle A\rangle B\ ) \vee (\ \langle B\rangle A\ )$	$\top$	$\langle A\rangle \top$	$\Rightarrow$	A
$[\,]$	$[A]B$	$\Rightarrow$	$\neg A \vee (\ \langle A\rangle B\ )$	$\top$	$\langle \top\rangle A$	$\Rightarrow$	A
$\rightarrow$	$A \rightarrow B$	$\Rightarrow$	$\neg A \vee B$				
$\leftrightarrow$	$A \leftrightarrow B$	$\Rightarrow$	$(\ \langle A\rangle B\ ) \vee (\ \langle \neg A\rangle \neg B\ )$	$\forall$	$\forall x\, P(x)$	$\Rightarrow$	$\langle\ \forall x\, P(x)\ \rangle\ P(t)$

Proof Assume that $\Gamma_{\langle A\rangle B}$ fails strongly in I. This means that the first formula F in $\Gamma_{\langle A\rangle B}$, for which $I(F) \neq \mathbf{t}$, has $I(F) = \mathbf{f}$.

- If F is among the Γ_i, then it is immediate that $\Gamma_{A,B}$ fails strongly in I.
- If $F = \langle A\rangle B$, then either $I(A) = \mathbf{f}$, or ($I(A) = \mathbf{t}$ and $I(B) = \mathbf{f}$). Since $I(\Gamma_1) = \cdots = I(\Gamma_m) = \mathbf{t}$, in both cases $\Gamma_{A,B}$ fails strongly in I.
- If F is among the Γ'_j, then F also occurs in $\Gamma_{A,B}$. We know that $I(\Gamma_1) = \cdots = I(\Gamma_m) = \mathbf{t}$. From the fact that $I(\ \langle A\rangle B\ ) = \mathbf{t}$, follows that $I(A) = I(B) = \mathbf{t}$. Since we assumed that $I(\Gamma'_1) = \cdots = I(\Gamma'_{j-1}) = \mathbf{t}$, it follows that F is the first formula in $\Gamma_{A,B}$ for which $I(F) \neq \mathbf{t}$. Since $I(F) = \mathbf{f}$, we know that $\Gamma_{A,B}$ fails strongly in I.

For the other direction, assume that $\Gamma_{A,B}$ fails strongly in I. Let F be the first formula in $\Gamma_{A,B}$ for which $I(F) \neq \mathbf{t}$. We have $I(F) = \mathbf{f}$.

- If F is among the Γ_i, then it is immediate that $\Gamma_{\langle A\rangle B}$ fails strongly in I.
- If $F = A$, then $I(\ \langle A\rangle B\ ) = \mathbf{f}$, and $I(\Gamma_1) = \cdots = I(\Gamma_m) = \mathbf{t}$, so that $\Gamma_{\langle A\rangle B}$ fails strongly in I at formula A.
- If $F = B$, then we know that $I(A) = \mathbf{t}$, so that $I(\ \langle A\rangle B\ ) = \mathbf{f}$. Since $I(\Gamma_1) = \cdots I(\Gamma_m) = \mathbf{t}$, it follows that $\Gamma_{\langle A\rangle B}$ fails strongly in I at formula B.
- If F is among the Γ'_j, then F also occurs in $\Gamma_{\langle A\rangle B}$, so that it is sufficient to show that there is no formula F' before F in $\Gamma_{\langle A\rangle B}$, s.t. $I(F') \neq \mathbf{t}$. The only candidate is $\langle A\rangle B$, because all other formulas were copied from $\Gamma_{A,B}$. But since we know that $I(A) = I(B) = \mathbf{t}$, it follows that $I(\ \langle A\rangle B\ ) = \mathbf{t}$.

Theorem 5 *Let $\Gamma_\exists$ be a sequent of form $\Gamma_1, \ldots, \Gamma_m,\ \exists x\ P(x),\ \Gamma'_1, \ldots, \Gamma'_n \vdash$. Let Γ_x be the sequent $\Gamma_x = \Gamma_1, \ldots, \Gamma_m,\ P(x),\ \Gamma'_1, \ldots, \Gamma'_n \vdash$. Assume that x is not free in any of the formulas $\Gamma_1, \ldots, \Gamma_m, \Gamma'_1, \ldots, \Gamma'_n$.*

Let $I = (D, \mathbf{f}, \mathbf{t}, \sqsubseteq, [\ ])$ be an arbitrary interpretation. Then $\Gamma_\exists$ fails strongly in I iff for every $d \in D$, the sequent Γ_x fails strongly in $I^x_d = (D, \mathbf{f}, \mathbf{t}, \sqsubseteq, [\ ]^x_d)$.

Proof In the proof, we make use of the fact that $I(\Gamma_i) = I^x_d(\Gamma_i)$ and $I(\Gamma'_j) = I^x_d(\Gamma'_j)$, because x is not free in Γ_i, Γ'_j.

Assume that $\Gamma_\exists$ fails strongly in I. This means that for the first formula F in $\Gamma_\exists$ with $I(F) \neq \mathbf{t}$, one has $I(F) = \mathbf{f}$.

- If F is one of the Γ_i, then $I(\Gamma_i) = I^x_d(\Gamma_i) = \mathbf{f}$, and for all i', $(1 \leq i' \leq i)$, $I(\Gamma_{i'}) = I^x_d(\Gamma_{i'}) = \mathbf{t}$, so that Γ_x fails strongly at Γ_i in every I^x_d.
- If F is $\exists x\, P(x)$, then the fact that $I(\ \exists x\, P(x)\ ) = \mathbf{f}$, implies that for every $d \in D$, $I^x_d(\ P(x)\ ) = \mathbf{f}$. Since for every i, $I(\Gamma_i) = I^x_d(\Gamma_i) = \mathbf{t}$, it follows that Γ_x fails strongly at formula $P(x)$ in every interpretation I^x_d.
- If F is one of the Γ'_j, then we know that for every $d \in D$, $I(\Gamma_1) = I^x_d(\Gamma_1) = \cdots = I(\Gamma_m) = I^x_d(\Gamma_m) = \mathbf{t}$. It can be seen from Definition 4 that
  $I(\ \exists x\, P(x)\ ) = \mathbf{t}$ implies that for every $d \in D$, either $I^x_d(\ P(x)\ ) = \mathbf{f}$, or $I^x_d(\ P(x)\ ) = \mathbf{t}$. If $I^x_d(\ P(x)\ ) = \mathbf{f}$, then Γ_x strongly fails at $P(x)$ in I^x_d. Otherwise, we have $I^x_d(\ P(x)\ ) = \mathbf{t}$ and $I(\Gamma'_1) = I^x_d(\Gamma'_1) = \cdots = I(\Gamma'_{j-1}) = I^x_d(\Gamma'_{j-1}) = \mathbf{t}$, and $I(\Gamma_j) = I^x_d(\Gamma_j) = \mathbf{f}$, so that Γ_x fails strongly at Γ'_j in I^x_d

For the other direction, we use contraposition, so assume that $\Gamma_\exists$ does not fail strongly in I. We show that there exists a $d \in D$, s.t. Γ_x does not fail strongly in I^x_d. We distinguish the following cases:

- The first formula F with $I(F) \neq \mathbf{t}$ is among the Γ_i and $I(\Gamma_i) \neq \mathbf{f}$. Since for all i, $(1 \leq i \leq m)$, $I(\Gamma_i) = I^x_d(\Gamma_i)$, the sequent Γ_x does not fail strongly in any I^x_d.

- The first formula F with $I(F) \neq \mathbf{t}$ is $\exists x\ P(x)$, and $I(\ \exists x\ P(x)\ ) \neq \mathbf{f}$. It follows from Definition 4 that there is a $d \in D$, s.t. $I^x_d(\ P(x)\ ) \notin \{\mathbf{f},\mathbf{t}\}$. In the corresponding I^x_d, the sequent Γ_x does not fail strongly.
- The first formula F for which $I(F) \neq \mathbf{t}$ is among the Γ'_j, and $I(\Gamma'_j) \neq \mathbf{f}$. Since $I(\ \exists x P(x)\ ) = \mathbf{t}$, there exists a $d \in D$, s.t. $I^x_d(\ P(x)\ ) = \mathbf{t}$. In I^x_d, we have $I(\Gamma_1) = I^x_d(\Gamma_1) = \cdots = I(\Gamma_m) = I^x_d(\Gamma_m) = \mathbf{t}$, and $I(\Gamma'_1) = I^x_d(\Gamma'_1) = \cdots = I(\Gamma'_{j-1}) = I^x_d(\Gamma'_{j-1}) = \mathbf{t}$, so that Γ_x does not fail strongly in I^x_d.
- There is no formula F for which $I(F) \neq \mathbf{t}$ in $\Gamma_\exists$. This case is analogeous to the previous case.

Theorem 6 *Let $\Gamma_{A\vee B}$ be a sequent of form $\Gamma_1,\ldots,\Gamma_m,A\vee B,\Gamma'_1,\ldots,\Gamma'_n\vdash$. Let $\Gamma_A = \Gamma_1,\ldots,\Gamma_m,A,\Gamma'_1,\ldots,\Gamma'_n\vdash$, and let $\Gamma_B = \Gamma_1,\ldots,\Gamma_m,B,\Gamma'_1,\ldots,\Gamma'_n\vdash$. Let I be an interpretation. The sequent $\Gamma_{A\vee B}$ fails strongly in I iff both of Γ_A and Γ_B fail strongly in I.*

Proof Assume that $\Gamma_{A\vee B}$ fails strongly in I. We have to show that Γ_A and Γ_B fail strongly in I. There is a formula $F \in \Gamma_{A\vee B}$, such that $[F] = \mathbf{f}$, and for all formulas $F' \in \Gamma_{A\vee B}$ occurring before F, $[F'] = \mathbf{t}$. We distinguish three cases, dependent on the position of F :

- F is among the Γ_i. It follows immediately that Γ_A and Γ_B fail strongly in I.
- F is $A\vee B$. It follows from $[A\vee B] = \mathbf{f}$, that $[A] = \mathbf{f}$, and $[B] = \mathbf{f}$. Since $[\Gamma_1] = \cdots = [\Gamma_m] = \mathbf{t}$, both Γ_A and Γ_B fail strongly in I.
- F is among the Γ'_j. Since F is the first formula in $\Gamma_{A\vee B}$, for which $[F] = \mathbf{f}$, we know that $[\Gamma_1] = \cdots = [\Gamma_m] = \mathbf{t}$, and $[A\vee B] = \mathbf{t}$. If $[A] = [B] = \mathbf{t}$, then both Γ_A and Γ_B fail strongly in I at formula F.
  If $[A] = \mathbf{f}$, and $[B] = \mathbf{t}$, then Γ_A fails strongly in I (at formula A), and Γ_B fails strongly at formula F.
  If $[A] = \mathbf{t}$, and $[B] = \mathbf{f}$, then Γ_A fails strongly in I at formula F, and Γ_B fails strongly in I at formula B.
  In can be checked from Definition 4, that there are no other cases where $[A\vee B] = \mathbf{t}$.

The easiest way to prove the other direction is by contraposition. Assume that $\Gamma_{A\vee B}$ does not fail strongly in I. We show that one of Γ_A,Γ_B does not fail strongly in I. The following cases can be distinguished:

- For all $F \in \Gamma_{A\vee B}$, we have $[F] = \mathbf{t}$. Then, for all i, $[\Gamma_i] = \mathbf{t}$, $[A\vee B] = \mathbf{t}$, and for all j, $[\Gamma'_j] = \mathbf{t}$. It can be seen from Definition 4, that the fact that $[A\vee B] = \mathbf{t}$, implies that either $[A] = \mathbf{t}$, or $[B] = \mathbf{t}$. In the first case, Γ_A does not fail in I. In the second case, Γ_B does not fail in I.
- The first $F \in \Gamma_{A\vee B}$, for which $[F] \neq \mathbf{t}$, has also $[F] \neq \mathbf{f}$, and F is among the Γ_i. In that case, both of Γ_A and Γ_B do not fail strongly in I.
- The first $F \in \Gamma_{A\vee B}$, for which $[F] \neq \mathbf{t}$, is $A\vee B$, and we also have $[A\vee B] \neq \mathbf{f}$. Either $[A] \notin \{\mathbf{f},\mathbf{t}\}$ or $[B] \notin \{\mathbf{f},\mathbf{t}\}$. Since for all i, $[\Gamma_i] = \mathbf{t}$, the first case implies that Γ_A does not fail strongly in I. The second case implies that Γ_B does not fail strongly in I.
- The first $F \in \Gamma_{A\vee B}$, for which $[F] \neq \mathbf{t}$, is among the Γ'_j, and also $[F] \neq \mathbf{f}$. Then for all i, $[\Gamma_i] = \mathbf{t}$. Also $[A\vee B] = \mathbf{t}$, which implies that either $[A] = \mathbf{t}$, or $[B] = \mathbf{t}$. In the first case, F is the first formula in Γ_A, for which $[F] \neq \mathbf{t}$, so that Γ_A does not fail strongly in I. In the second case, F is the first formula in Γ_B, for which $[F] \neq \mathbf{t}$, so that Γ_B does not fail strongly in I.

5 Completeness

In Section 4 we introduced Seq_{PCL} and proved its soundness. We will now give an outline of the completeness proof. If the $\forall$-quantifier would not exist, then we would already now have a completeness proof. Seq_{PCL} has sufficiently many rules that preserve strong unsatisfiability in both directions. This means that the conclusion of the rule is strongly unsatisfiable if and only if all of its premises are strongly unsatisfiable. If some sequent is not strongly unsatisfiable, then one can find a rule that could introduce the sequent, but for which one of the premises is not strongly unsatisfiable. By continuing, it is possible to reduce the original sequent to a sequent that contains only possible negations of possible Props of atoms. These simple sequents are either directly derivable from axioms by equality reasoning, or it is easy to construct an interpretation in which they do not fail strongly. By the equivalence property, this implies that we can construct either a proof of the original sequent, or a counter interpretation.

In order to include $\forall$ in the completeness proof, we would like to proceed in a standard way: First allow each $\forall$-quantifier to have some fixed set of instances. If no proof can be constructed, then grant each $\forall$-quantifier one instance more. This process either results in a proof, or it leads to an increasing sequence of simple sequents from which an interpretation can be read off in the limit.

Unfortunately, there is a problem with this approach, which is caused by the fact that in most cases the limit will be infinite. We want to show that the limit sequent does not fail strongly in the limit interpretation I (and that none of the sequents on the way fails strongly in I), but we have no concept of strong failure for infinite sequents. One possible solution would be to introduce a proper notion, based on graph sequents, but we don't want to do that here. It turns out that there is a simpler approach, which avoids introducing special notions for infinite sequents:

Definition 8 Let $\Gamma = \Gamma_1, \ldots, \Gamma_n \vdash$ be a sequent. We say that Γ is *in* Prop *normal form* (PNF) if for every Γ_i, either **(1)** Γ_i is of form $t_1 = t_2$, $t_1 \neq t_2$, $\text{Prop}(A)$ or $\neg\text{Prop}(A)$, or **(2)** there is a $j < i$, s.t. Γ_j has form $\text{Prop}(\Gamma_i)$.

Lemma 1 *Let $\Gamma \vdash$ be a sequent in PNF. Let I be an interpretation. Then $\Gamma \vdash$ does not fail strongly in I iff for every formula F in Γ, $I(F) = \mathbf{t}$.*

Theorem 7 *If* Seq_{PCL} *is complete for sequents in PNF, then it is complete for all sequents.*

Proof Assume that Seq_{PCL} is complete for sequents in PNF. Let $\Gamma \vdash$ be an arbitrary sequent. Write $\Gamma \vdash$ in the form $\Gamma_1, \ldots, \Gamma_n \vdash$. Let $\#\Gamma \vdash$ be the number of violations of Definition 8 in $\Gamma \vdash$, i.e. the number of Γ_i that are not of form $t_1 = t_2$, $t_1 \neq t_2$, $\text{Prop}(A)$, $\neg\text{Prop}(A)$, and for which there also exists no $j < i$ with $\Gamma_j = \text{Prop}(\Gamma_i)$. We proceed by induction on $\#\Gamma \vdash$.

If $\#\Gamma \vdash = 0$, then $\Gamma \vdash$ is in PNF, so that we are done. Otherwise, assume that the first violation of Definition 8 occurs on position i. This implies that the sequent $S_1 = \Gamma_1, \ldots, \Gamma_{i-1}, \neg\text{Prop}(\Gamma_i) \vdash$ is in PNF. If S_1 has no proof, then by PNF-completeness, we know that there exists an interpretation I, in which S_1 does not fail strongly. By Lemma 1, $I(\Gamma_1) = \cdots = I(\Gamma_{i-1}) = I(\ \neg\text{Prop}(\Gamma_i)\ ) = \mathbf{t}$, so that $I(\ \text{Prop}(\Gamma_i)\ ) = \mathbf{f}$. This implies that the sequent $\Gamma_1, \ldots, \Gamma_{i-1}, \Gamma_i, \ldots, \Gamma_n \vdash$ fails in I, but not strongly. As a consequence, we have completeness for this case.

If S_1 does have a proof, then we consider the sequent
$S_2 = \Gamma_1, \ldots, \Gamma_{i-1}, \text{Prop}(\Gamma_i), \Gamma_i, \ldots, \Gamma_n \vdash$. Clearly, $\#S_2 = (\#\Gamma \vdash) - 1$, so that we can assume completeness for S_2.

If S_2 has no proof, then there exists an interpretation I, in which S_2 does either not fail at all, or it fails but not strongly. If S_2 does not fail in I, then $\Gamma \vdash$ also does not fail, and we are done. Otherwise, consider the first formula F in S_2, for which $I(F) \neq \mathbf{t}$. If F were among the $\Gamma_1, \ldots, \Gamma_{i-1}$, this would imply that the sequent S_2 fails strongly, due to the fact that S_1 has a proof. From the provability of S_1 follows, that F cannot be $\mathrm{Prop}(\Gamma_i)$. If F would be Γ_i, this would imply that $I(\ \mathrm{Prop}(\Gamma_i)\ ) = \mathbf{f}$, which contradicts the fact that S_2 is provable. So it must be the case that F is among $\Gamma_{i+1}, \ldots, \Gamma_n$. But this implies that $\Gamma \vdash$ also fails non strongly in I, so that we have completeness in this case as well.

Finally assume that S_2 has a proof. In that case, we can combine the proofs of S_1 and S_2 into a proof of $\Gamma \vdash$ as follows:

$$\frac{\Gamma_1, \ldots, \Gamma_{i-1},\ \neg\mathrm{Prop}(\Gamma_i) \vdash \qquad \Gamma_1, \ldots, \Gamma_{i-1}, \mathrm{Prop}(\Gamma_i), \Gamma_i, \ldots, \Gamma_n \vdash}{\Gamma_1, \ldots, \Gamma_{i-1}, \Gamma_i, \ldots, \Gamma_n \vdash} \ (\mathit{cut} + \mathit{weakening}).$$

The fact that we can restrict our attention to sequents in PNF, simplifies the completeness proof quite a lot. By Lemma 1, we know that we are looking either for a proof, or an interpretation that makes all atoms in the sequent true. Since this does not rely on order anymore, we can use standard techniques to construct the limit of the sequents in the failed proof attempt. We still have to show two things, but they turn out unproblematic: **(1)** It does not happen that, during proof search for a sequent in PNF, one needs to make use of a sequent that is not in PNF. **(2)** All formulas in the original sequent are true in the resulting interpretation. In order to do this, we show that a nonsucceeding proof attempt converges towards a saturated set, which is defined as follows:

Definition 9 Let Σ be a set of formulas. We call Σ *saturated* if it has the following properties:

- If $A \in \Sigma$, and A is not of form $t_1 = t_2$, $t_1 \neq t_2$, $\mathrm{Prop}(B)$, or $\neg\mathrm{Prop}(B)$, then $\mathrm{Prop}(A) \in \Sigma$.
- $\bot \notin \Sigma$.
- There exist no terms t, u, no $n \geq 0$, no sequence of terms $t_1, u_1, \ldots, t_n, u_n$, s.t. $\{t_1 = u_1, \ldots, t_n = u_n, t \neq u\} \subseteq \Sigma$, and $t_1 = u_1, \ldots, t_n = u_n \vdash t = u$ in the standard theory of equality.
- There exist no atoms A, A', no $n \geq 0$, no sequence of terms $t_1, u_1, \ldots, t_n, u_n$, s.t. $\{A,\ t_1 = u_1, \ldots, t_n = u_n,\ \neg A'\} \subseteq \Sigma$, and $A,\ t_1 = u_1, \ldots, t_n = u_n \vdash A'$ in the standard theory of equality.
- If $\{\ \mathrm{Prop}(A \vee B),\ A \vee B\ \} \subseteq \Sigma$, then either $\{\mathrm{Prop}(A), A\} \subseteq \Sigma$, or $\{\mathrm{Prop}(B), B\} \subseteq \Sigma$.
- If $\{\ \mathrm{Prop}(\ \langle A \rangle B\ ),\ \langle A \rangle B\ \} \subseteq \Sigma$, then $\{\mathrm{Prop}(A), \mathrm{Prop}(B), A, B\} \subseteq \Sigma$.
- If $\{\ \mathrm{Prop}(\ \exists x\, P(x)\ ),\ \exists x\, P(x)\ \} \subseteq \Sigma$, then there exists a term t, s.t. $\{\ \mathrm{Prop}(\ P(t)\ ),\ P(t)\ \} \subseteq \Sigma$.
- If $\{\ \mathrm{Prop}(\ \forall x\, P(x)\ ),\ \forall x\, P(x)\ \} \subseteq \Sigma$, then for every term t that can be formed from the signature of Σ, we have $\{\ \mathrm{Prop}(P(t)),\ P(t)\ \} \subseteq \Sigma$.
- For every instance $A \Rightarrow B$ of a rule in Figure 1 or Figure 4, if $\{\mathrm{Prop}(A), A\} \subseteq \Sigma$, then $\{\mathrm{Prop}(B), B\} \subseteq \Sigma$.
- For every instance $\mathrm{Prop}(A) \Rightarrow B$ of a rule in Figure 2, if $\mathrm{Prop}(A) \in \Sigma$, but $A \notin \Sigma$, then $B \in \Sigma$.

Note that, by taking $n = 0$ in the fourth case, the definition of saturated set implies that Σ does not contain a complementary pair of atoms $A, \neg A$. Since our aim is to prove completeness, we need a proof search strategy that converges towards a saturated set in the limit. In order to obtain a saturated set, it is necessary to preserve PNF during proof search. If, for

Fig. 5 Preservation of PNF under $\langle\,\rangle$-intro

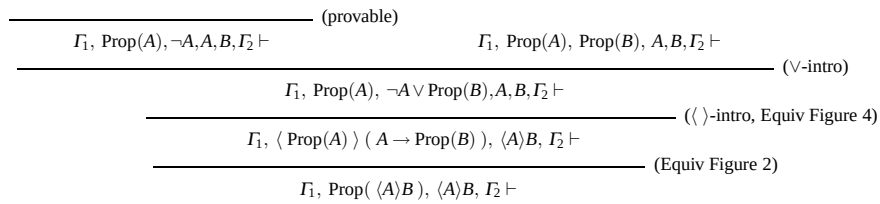


Fig. 6 Preservation of PNF under $\vee$-intro

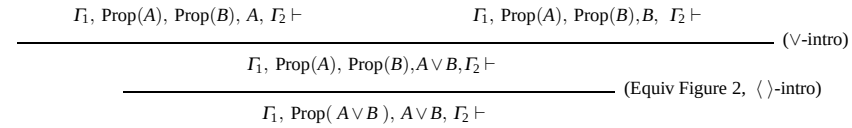
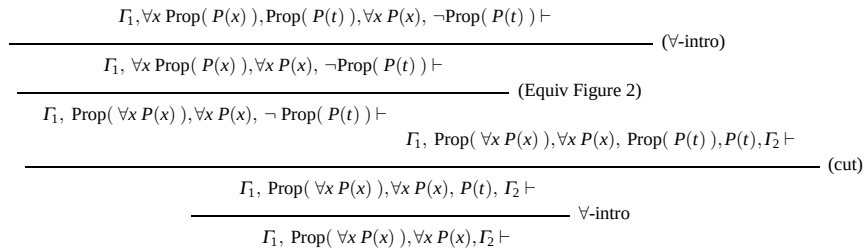


Fig. 7 Preservation of PNF under $\forall$-intro



example, one has a sequent of form $\Gamma_1, \mathrm{Prop}(A \wedge B), A \wedge B, \Gamma_2 \vdash$ and tries to prove it from $\Gamma_1, \mathrm{Prop}(A \wedge B), A, B, \Gamma_2 \vdash$, then the new sequent is not in PNF anymore. In this case, we can continue proof search by replacing $\mathrm{Prop}(A \wedge B)$ by $\langle \mathrm{Prop}(A) \rangle \mathrm{Prop}(B)$, which in turn can be replaced by $\mathrm{Prop}(A), \mathrm{Prop}(B)$, which is in PNF again. Figures 5, 6, 7 and 8 show that, for the operators $\langle\,\rangle, \vee, \forall$ and $\exists$, it is always possible to continue proof search with sequents in PNF. All of the remaining cases can be reduced to the cases for $\langle\,\rangle$ and $\vee$, using the equivalences in Figures 1, 2 and 4.

In Figure 5, the leftmost sequent is provable, because it is in PNF, and it contains the complementary pair $A, \neg A$. Hence, it is sufficient to continue proof search with the rightmost sequent, which is also in PNF.

Figure 7 shows how PNF can be preserved when instantiating a $\forall$. In the middle, the proof splits at the cut application. The first formula of the left branch is provable, because it contains the complementary pair $\mathrm{Prop}(P(t)), \neg \mathrm{Prop}(P(t))$, Γ_1 is in PNF, and the remaining formulas $\forall x\ \mathrm{Prop}(\ P(x)\ )$, $\mathrm{Prop}(P(t))$, $\forall x\ P(x)$ can be easily proven Prop in their respective contexts. The right premise of the cut application is in PNF, and has the formulas $\mathrm{Prop}(P(t)), P(t)$ added. Figure 8 branches at the weakening step, and its first premise is provable.

Fig. 8 Preservation of PNF under ∃-intro

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma_1,\ \neg\mathrm{Prop}(\ \forall x\ \mathrm{Prop}(\ P(x)\ )\vdash \qquad\qquad \Gamma_1,\ \mathrm{Prop}(\ P(y)\ ),\ P(y),\ \Gamma_2\vdash}{\Gamma_1,\ \forall x\ \mathrm{Prop}(\ P(x)\ ),\ \mathrm{Prop}(\ P(y)\ ),\ P(y),\ \Gamma_2\vdash}\ \text{(weakening)}}{\Gamma_1,\ \forall x\ \mathrm{Prop}(\ P(x)\ ),\ P(y),\ \Gamma_2\vdash}\ (\forall\text{-intro})}{\Gamma_1,\ \forall x\ \mathrm{Prop}(\ P(x)\ ),\exists x\ P(x),\ \Gamma_2\vdash}\ (\exists\text{-intro})}{\Gamma_1,\ \mathrm{Prop}(\ \exists x\ P(x)\ ),\exists x\ P(x),\Gamma_2\vdash}\ \text{(Equiv Figure 2)}$$

It remains to extract an interpretation $I_\Sigma = (D_\Sigma, \mathbf{f}_\Sigma, \mathbf{t}_\Sigma, \sqsubseteq_\Sigma, [\ ]_\Sigma)$ from the saturated set Σ. This can be done as follows:

- Assume two designated objects $\mathbf{f}, \mathbf{t}$ that are not in the signature of Σ. They will represent the truth values. Let T_Σ be the set of terms that can be formed from the signature of Σ. Let $\equiv$ be the smallest congruence relation on $T_\Sigma \cup \{\mathbf{f}, \mathbf{t}\}$, s.t.
  - for all $t_1, t_2 \in T_\Sigma$, if $(t_1 = t_2) \in \Sigma$, then $t_1 \equiv t_2$,
  - for all $t \in T_\Sigma$, if both of $\mathrm{Prop}(t), t \in \Sigma$, then $t \equiv \mathbf{t}$,
  - for all $t \in T_\Sigma$, if $\mathrm{Prop}(t) \in \Sigma$, but $t \notin \Sigma$, then $t \equiv \mathbf{f}$.

  The domain D_Σ of I_Σ is defined as $(T \cup \{\mathbf{f}, \mathbf{t}\})/\equiv$.
- $\mathbf{f}_\Sigma$ is the element of D_Σ that contains $\mathbf{f}$.
- $\mathbf{t}_\Sigma$ is the element of D_Σ that contains $\mathbf{t}$.
- The choice of $\sqsubseteq_\Sigma$ is not important, so we simply select an arbitrary total order on $D_\Sigma \backslash \{\mathbf{f}_\Sigma, \mathbf{t}_\Sigma\}$.
- The function $[\ ]_\Sigma$ is defined in such a way that for every $t \in T_\Sigma$, the interpretation $I_\Sigma(t)$ is the equivalence class of $\equiv$ in which t falls.

It remains to show that for every formula A, $A \in \Sigma \Leftrightarrow I_\Sigma(A) = \mathbf{t}$. This can be proven by structural induction on A using the properties in Definition 9.

Theorem 8 *Sequent calculus* Seq_{PCL} *is complete: If a sequent* $\Gamma \vdash$ *is not provable in* Seq_{PCL}, *then there exists an interpretation* I *in which* $\Gamma \vdash$ *does not fail strongly.*

6 A Practical Calculus: Seq^2_{PCL}

We have shown that Seq_{PCL} is sound and complete for strong unsatisfiability, but unfortunately Seq_{PCL} is not suitable for practical use, mostly because it is substructural. We will introduce another calculus, which we call Seq^2_{PCL}, and which is suitable for practical use. Its name is derived from the fact that its sequents consist of two components, a first component that has already been type checked, and a second component which has not yet been typechecked.

The first component of a Seq^2_{PCL} sequent is similar to a Seq_{PCL} sequent: The order of formulas cannot be changed, and weakening is restricted. The second component behaves similar to standard sequent calculus for classical logic. It has all usual structural rules. This makes Seq^2_{PCL} usable in practice.

One of the problems of Seq_{PCL} is its absence of structural rules. The order of formulas in a sequent cannot be changed, while at the same time the reasoning rules operate at arbitrary positions in the sequent. This makes printing of sequent proofs with many premises

Fig. 9 Problems with cut in Seq_{PCL} for $\vee$ and $\wedge$

$$\dfrac{\dfrac{\Gamma_1, A, \Gamma_2 \vdash \qquad \Gamma_1, B, \Gamma_2 \vdash}{\Gamma_1, A \vee B, \Gamma_2 \vdash} \qquad \dfrac{\Gamma_1, \neg A, \neg B, \Gamma_2 \vdash \qquad \Gamma_1, \neg B, \neg A, \Gamma_2 \vdash}{\Gamma_1, \neg A \wedge \neg B, \Gamma_2 \vdash}}{\Gamma_1, \neg(A \vee B), \Gamma_2 \vdash} \qquad \dfrac{\Gamma_1, \neg(A \vee B), \Gamma_2 \vdash}{\Gamma_1, \Gamma_2 \vdash}$$

impossible. If a rule changes a formula in the middle of a sequent, then all formulas that come after it have to be reprinted.

Another practical disadvantage of Seq_{PCL} is the fact that it mixes type checking with proving, which is due to the fact that it preserves strong unsatisfiability and this notion mixes type correctness with validity. If one for example tries to prove the sequent $\mathrm{Prop}(A \wedge B),\ A \wedge B \wedge C,\ \neg A \vee \neg B \vee \neg C \vdash$ by backward search, then one notices only after a lot of branching that the sequent $\mathrm{Prop}(A),\ \mathrm{Prop}(B), A, B, C,\ \neg C \vdash$ is not provable, because C is not declared Prop.

In Seq^2_{PCL}, type (and precondition) checking is separated from proving: A formula is typechecked at the moment it is moved from the second part of the sequent to the first part. Proving always takes place in the second part of the sequent.

The third and most important problem with Seq_{PCL} is the fact that it does not allow a practical form of cut elimination. Cut elimination is important for the automatic generation of proofs by theorem provers, because it cleans up redundancies that are created when subformula replacements inside a formula are chained. This happens for example, when $A \leftrightarrow B$ is first replaced by $(A \rightarrow B) \wedge (B \rightarrow A)$, and after that, by $(\neg A \vee B) \wedge (\neg B \vee A)$ in some formula context $F[\,]$.

Unfortunately, cut elimination in Seq_{PCL} is problematic. The problem can be seen from the proof in Figure 9. It consists of an $\vee$-introduction and a $\wedge$-introduction, followed by a cut on the introduced formulas. In order to permute the cut downwards, one would have to cut $\Gamma_1, A, \Gamma_2 \vdash$ with $\Gamma_1,\ \neg A,\ \neg B,\ \Gamma_2 \vdash$, which would result in $\Gamma_1, \neg B, \Gamma_2 \vdash$. In order to do this, $\Gamma_1, A, \Gamma_2 \vdash$ has to be weakened into $\Gamma_1, A, \neg B, \Gamma_2 \vdash$, which requires a proof of the sequent $\Gamma_1, A, \neg\mathrm{Prop}(\neg B) \vdash$. It can be proven that a sequent of this form is provable in general, but it is very tedious to have to provide additional type information, every time when a cut is permuted.

We now introduce Seq^2_{PCL}. Sequents of Seq^2_{PCL} consist of two parts. The first component of the sequent has essentially the same semantics as a Seq_{PCL} sequent. The second component of the sequent is a multiset with a semantics that is similar to the semantics of a classical sequent.

Definition 10 A Seq^2_{PCL} *sequent* is an object of form $\Gamma_1; \Gamma_2 \vdash$ where Γ_1 is a sequence of formulas, and Γ_2 is a multiset of formulas.

For the multiset Γ_2 we say that Γ_2 *fails in* I if there is an $A \in \Gamma_2$, s.t. $I(A) \neq \mathbf{t}$.

We say that a two-level sequent $\Gamma_1; \Gamma_2 \vdash$ *fails in an interpretation* I if either Γ_1 or Γ_2 fails in I.

We say that $\Gamma_1; \Gamma_2 \vdash$ *fails strongly* in I, if either $\Gamma_1 \vdash$ fails strongly in I, or Γ_1 does not fail in I and Γ_2 fails in I.

We call $\Gamma_1; \Gamma_2 \vdash$ *unsatisfiable* if $\Gamma_1; \Gamma_2 \vdash$ fails in every interpretation. We call $\Gamma_1; \Gamma_2 \vdash$ *strongly unsatisfiable* if $\Gamma_1; \Gamma_2 \vdash$ fails strongly in every interpretation.

Fig. 10 Logical Rules of Seq^2_{PCL} :

Rules for $\wedge$ and $\vee$

$$\frac{\Gamma_1;\Gamma_2,A,B\vdash}{\Gamma_1;\Gamma_2,A\wedge B\vdash} \qquad \frac{\Gamma_1;\Gamma_2,A,\mathrm{Prop}(B)\vdash \qquad \Gamma_1;\Gamma_2,B,\mathrm{Prop}(A)\vdash}{\Gamma_1;\Gamma_2,\ A\vee B\vdash}$$

Rules for $\langle\ \rangle$ and $[\,]$

$$\frac{\Gamma_1;\Gamma_2,A,B\vdash}{\Gamma_1;\Gamma_2,\langle A\rangle B\vdash} \qquad \frac{\Gamma_1;\Gamma_2,\neg A\vdash \qquad \Gamma_1;\Gamma_2,\mathrm{Prop}(A),B\vdash}{\Gamma_1;\Gamma_2,\ [A]B\vdash}$$

Rules for $\forall$ and $\exists$

$$\frac{\Gamma_1;\Gamma_2,\ P(t)\vdash}{\Gamma_1;\Gamma_2,\ \forall x\,P(x)\vdash} \qquad \frac{\Gamma_1;\Gamma_2,\ P(y)\vdash}{\Gamma_1;\Gamma_2,\ \exists x\,P(x)\vdash}$$

(In the $\forall$-rule, t must be a term. In the $\exists$-rule, y must be not free in Γ_1 or Γ_2.)

Equivalence If $A\Rightarrow B$ is an instance of one of the rules in Figure 1, Figure 2 or the rule $(P\leftrightarrow Q)\Rightarrow(\neg P\vee Q)\wedge(P\vee\neg Q)$, then the following derivation is possible:

$$\frac{\Gamma_1;\Gamma_2,B\vdash}{\Gamma_1;\Gamma_2,A\vdash}$$

Equality Replacement

$$\frac{\Gamma_1;\Gamma_2,A[t_1],\ t_1=t_2\vdash}{\Gamma_1;\Gamma_2,A[t_2],\ t_1=t_2\vdash}$$

(Replacement in the other direction is also allowed.)

The sequents $;A,\neg A\vdash$ and $;\neg(A\vee B),A\vdash$ are strongly unsatisfiable, while the sequents $A,\neg A;\vdash$ and $A;\neg(A\vee B)\vdash$ are not.

Sequent calculus Seq^2_{PCL} inherits the reduction rules of Figure 1 and Figure 2 from Seq_{PCL}. In addition, it has the reduction rule $A\leftrightarrow B\ \Rightarrow(\neg A\vee B)\wedge(A\vee\neg B)$. The logical rules of Seq^2_{PCL} are given in Figure 10. Axioms and structural rules are given in Figure 11, and Figure 12 contains the rules for building contexts.

A sequent of form $\Gamma_1;\bot\vdash$ represents the fact that Γ_1 is a well-typed context. The rules in Figure 12 are intended to built up well-typed contexts. The axioms in Figure 11 take a well-typed context as a starting point for further proof. The extension rule of Figure 12 makes it possible to separate proving from type checking, because a formula can only be moved to the first part of a sequent after it has been type checked. One can view Seq_{PCL} and Seq^2_{PCL} as exploiting the two different views on strong unsatisfiability, that are linked by Theorem 1.

In order to illustrate usage of Seq^2_{PCL}, we give a complete proof of a de Morgan formula. The equivalent sequent in simply typed logic would be

$$D{:}\,\mathrm{Type},\ A{:}\,D\to\mathrm{Prop}\vdash\neg\forall x{:}\,D\,A(x)\leftrightarrow\exists x{:}\,D\,\neg A(x).$$

Example 1 Suppose that we want to prove strong validity of the following sequent:

$$\forall x\,\mathrm{Prop}(D(x)),\ \forall x\,D(x)\to\mathrm{Prop}(A(x))\vdash\neg\forall x[D(x)]A(x)\leftrightarrow\exists x\langle D(x)\rangle\neg A(x).$$

In order to do this, we have to prove the following Seq^2_{PCL} sequent:

$$\forall x\,\mathrm{Prop}(D(x)),\ \forall x\,D(x)\to\mathrm{Prop}(A(x)),\ \neg(\neg\forall x[D(x)]A(x)\leftrightarrow\exists x\langle D(x)\rangle\neg A(x)\ );\vdash.$$

Fig. 11 Axioms and Structural Rules of Seq^2_{PCL} :
Axioms

$$\frac{}{\Gamma_1;\bot\vdash} \qquad \frac{}{\Gamma_1;\bot\vdash} \qquad \frac{}{\Gamma_1;\bot}$$
$$\Gamma_1;\Gamma_2,A,\neg A\vdash \qquad \Gamma_1;\Gamma_2,\ t\neq t\vdash \qquad \Gamma_1;\Gamma_2,\neg\mathrm{Prop}(A),\pm A\vdash$$

Weakening

$$\frac{\Gamma_1;\Gamma_2\vdash}{\Gamma_1;\Gamma_2,A\vdash}$$

(A can be positive or negative.)
Contraction

$$\frac{\Gamma_1,A,\Gamma_2;\Gamma_3,A\vdash}{\Gamma_1,A,\Gamma_2;\Gamma_3\vdash} \qquad \frac{\Gamma_1;\Gamma_2,A,A\vdash}{\Gamma_1;\Gamma_2,A\vdash}$$

(A can be positive or negative.)
Cut

$$\frac{\Gamma_1;\Gamma_2,\neg\mathrm{Prop}(A)\vdash \qquad \Gamma_1;\Gamma_2,\neg A\vdash \qquad \Gamma_1;\Gamma_2,A\vdash}{\Gamma_1;\Gamma_2\vdash}$$

Fig. 12 Context Rules of Seq^2_{PCL} :

Start

$$\frac{}{;\bot\vdash}$$

Extension

$$\frac{\Gamma_1;\neg\mathrm{Prop}(A)\vdash \qquad \Gamma_1;\Gamma_2,A\vdash}{\Gamma_1,A;\Gamma_2\vdash}$$

This can be done by using the Extension rule, so that we have to prove the following two sequents:

$$\forall x\ \mathrm{Prop}(D(x)),\ \forall x\ D(x)\to\mathrm{Prop}(A(x));\ \neg\mathrm{Prop}(\neg(\neg\forall x[D(x)]A(x)\leftrightarrow\exists x\langle D(x)\rangle\neg A(x)\ ))\vdash,$$

and

$$\forall x\ \mathrm{Prop}(D(x)),\ \forall x\ D(x)\to\mathrm{Prop}(A(x));\ \neg(\neg\forall x[D(x)]A(x)\leftrightarrow\neg\exists x\ \langle D(x)\rangle\neg A(x)\ )\vdash .$$

Both sequents are easily provable, but they reduce to axioms that require a proof of the sequent

$$\forall x\ \mathrm{Prop}(D(x)),\ \forall x\ D(x)\to\mathrm{Prop}(A(x));\ \bot\vdash .$$

This sequent has to be proven from

$$\forall x\ \mathrm{Prop}(D(x));\ \neg\mathrm{Prop}(\forall x\ D(x)\to\mathrm{Prop}(x))\vdash,$$

again by using Extension (the other premise has a trivial proof). The resulting axiom requires a proof of

$$\forall x\ \mathrm{Prop}(D(x));\bot\ \vdash,$$

which can be proven by another application of Extension.

It can be checked by case analysis that all rules of Seq^2_{PCL} preserve strong unsatisfiability. The cut rule in Figure 11 would be unsound without the first premise, because one could have an interpretation I, in which $I(A) \notin \{\mathbf{f}, \mathbf{t}\}$ and $I(\Gamma_1) = I(\Gamma_2) = \mathbf{t}$. The second and the third premise would fail strongly, while the conclusion would not fail at all.

Seq^2_{PCL} solves the problems of Seq_{PCL} that we mentioned at the beginning of this section. It clearly has separation of type checking and proving. The rules in Figure 10 and Figure 11 operate on the second component of the two-level sequent, which is a multiset. This ensures that the usual structural rules are available, so that formulas can be permuted freely. As a consequence, it is possible to present proofs in either semantic tableau style, or natural deduction style, both of which are concise and readable. We show in the next section that Seq^2_{PCL} has cut elimination.

7 Cut Elimination for Seq^2_{PCL}

We show that the cut rule in Figure 11 can be permuted downwards (in the direction of the axioms) through all logical rules in Figure 10, with the exception of equality replacement. By using standard permutations, equality replacement can also be permuted downwards to the axioms, so that it is possible to obtain a proof in which equality replacement is applied only on sequents that are axioms by themselves, or obtained by obtaining equality replacement on axioms. Cut can be permuted downward until it either reaches an axiom, or a sequent that is derived by equality replacement from an axiom.

In theory, cut elimination can cause a hyperexponential blow up of proofs, see [12] or [14]. In practice, it is easy to detect the cases where this would happen. Increase of proof size happens when a cut is permuted with a contraction. It can be easily checked that in the application that we have in mind, elimination of repeated buildups and breakdowns of contexts, no such permutations can occur.

In [5], cut elimination for sequent calculus in classical logic is described. Because Seq^2_{PCL} operates on multisets with the standard structural rules, the cut elimination procedure for Seq^2_{PCL} is similar to the procedure in [5]. In particular, cut can be always permuted with an introduction rule that does not introduce the cut formula. The fact that cut in Seq^2_{PCL} cuts away three formulas does not change this property. After these permutations, the only cases that need to be checked are the cases where a cut is applied on sequents, in each of which the cut formula is introduced directly before the cut. There are four possible cases, dependent on the logical operator being introduced. Three of the four possible cases are given in Figures 13, 14 and 15. We omitted the case where cut permutes with $\leftrightarrow$ introduction because it can be reduced to the $\wedge, \vee$-case by writing out the definition of $\leftrightarrow$.

8 Automatic Proofs of Propositional Rules

The propositional rules of both of the sequent calculi, and the propositional equivalences in Figures 1, 2 and Figure 4 can be checked by case analysis. Since there may be many cases, checking the cases by hand is tedious and it is easy to make mistakes. We sketch a general method by which such case analysis proofs can be checked automatically. All propositional sequent rules and all propositional equivalences have been checked in this way. The program[1] was written in C^{++}.

[1] The program can be downloaded from the author's homepage.

Fig. 13 Permutation of cut through $\wedge, \vee$-introduction

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma_1;\ \Gamma_2, \neg\mathrm{Prop}(A) \vdash \qquad \Gamma_1;\ \Gamma_2, \neg\mathrm{Prop}(B) \vdash}{\Gamma_1;\ \Gamma_2, \neg\mathrm{Prop}(A) \vee \neg\mathrm{Prop}(B) \vdash}}{\Gamma_1;\ \Gamma_2, \neg\mathrm{Prop}(A \wedge B) \vdash} \qquad \dfrac{\Gamma_1;\ \Gamma_2, A, B \vdash}{\Gamma_1;\ \Gamma_2, A \wedge B \vdash} \qquad \dfrac{\Gamma_1;\ \Gamma_2, \neg A \vdash \qquad \Gamma_1;\ \Gamma_2, \neg B \vdash}{\Gamma_1;\ \Gamma_2, \neg A \vee \neg B \vdash}}{\Gamma_1;\ \Gamma_2 \vdash}$$

$\Rightarrow$

$$\dfrac{\dfrac{\Gamma_1;\ \Gamma_2, \neg\mathrm{Prop}(A) \vdash \quad \Gamma_1;\ \Gamma_2, A, B \vdash \quad \Gamma_1;\ \Gamma_2, \neg A \vdash}{\Gamma_1;\ \Gamma_2, B \vdash} \qquad \Gamma_1;\ \Gamma_2, \neg B \vdash \qquad \Gamma_1;\ \Gamma_2, \neg\mathrm{Prop}(B) \vdash}{\Gamma_1;\ \Gamma_2 \vdash}$$

Fig. 14 Permutation of cut through $[\,], \langle\,\rangle$-introduction

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma_1;\ \Gamma_2, A, \neg\mathrm{Prop}(B) \vdash}{\Gamma_1;\ \Gamma_2, \mathrm{Prop}(\mathrm{Prop}(A)), A \wedge \neg\mathrm{Prop}(B) \vdash}}{\Gamma_1;\ \Gamma_2, \neg\mathrm{Prop}(A) \vdash} \quad \Gamma_1;\ \Gamma_2, [\mathrm{Prop}(A)](A \wedge \neg\mathrm{Prop}(B)) \vdash}{\Gamma_1;\ \Gamma_2, \neg\langle\mathrm{Prop}(A)\rangle(A \to \mathrm{Prop}(B)) \vdash}}{\Gamma_1;\ \Gamma_2, \neg\mathrm{Prop}([A]B) \vdash} \qquad \dfrac{\Gamma_1;\ \Gamma_2, \neg A \vdash \qquad \Gamma_1;\ \Gamma_2, \mathrm{Prop}(A), B \vdash}{\Gamma_1;\ \Gamma_2, [A]B \vdash} \qquad \dfrac{\Gamma_1;\ \Gamma_2, A, \neg B \vdash}{\Gamma_1;\ \Gamma_2, \langle A\rangle \neg B \vdash}}{\Gamma_1;\ \Gamma_2 \vdash}$$

$\Rightarrow$

$$\dfrac{\dfrac{\dfrac{\Gamma_1;\ \Gamma_2, \neg\mathrm{Prop}(A) \vdash \qquad \Gamma_1;\ \Gamma_2, \mathrm{Prop}(A), B \vdash}{\Gamma_1;\ \Gamma_2, B \vdash} \qquad \Gamma_1;\ \Gamma_2, A, \neg B \vdash \qquad \Gamma_1;\ \Gamma_2, A, \neg\mathrm{Prop}(B) \vdash}{\begin{array}{c}\vdots\\ \Gamma_1;\ \Gamma_2, A \vdash\end{array} \qquad \Gamma_1;\ \Gamma_2, \neg A \vdash \qquad \Gamma_1;\ \Gamma_2, \neg\mathrm{Prop}(A) \vdash}}{\Gamma_1;\ \Gamma_2 \vdash}$$

Fig. 15 Permutation of cut through $\forall, \exists$-introduction

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma_1;\ \Gamma_2, \neg\mathrm{Prop}(P(y)) \vdash}{\Gamma_1;\ \Gamma_2, \exists x\ \neg\mathrm{Prop}(P(x)) \vdash}}{\Gamma_1;\ \Gamma_2, \neg\mathrm{Prop}(\forall x\, P(x)) \vdash} \qquad \dfrac{\Gamma_1;\ \Gamma_2, P(t) \vdash}{\Gamma_1;\ \Gamma_2, \forall x\, P(x) \vdash} \qquad \dfrac{\Gamma_1;\ \Gamma_2, \neg P(y) \vdash}{\Gamma_1;\ \Gamma_2, \exists x\ \neg P(x) \vdash}}{\Gamma_1;\ \Gamma_2 \vdash}$$

$\Rightarrow$

$$\dfrac{\Gamma_1;\ \Gamma_2, P(t) \vdash \qquad \Gamma_1;\ \Gamma_2, \neg P(t) \vdash \qquad \Gamma_1;\ \Gamma_2, \neg\mathrm{Prop}(t) \vdash}{;\ \Gamma_1, \Gamma_2 \vdash}$$

In the permuted proof, the proofs of $\Gamma_1;\ \Gamma_2, \neg\mathrm{Prop}(P(y)) \vdash$ and $\Gamma_1;\ \Gamma_2, \neg P(y) \vdash$ have been instantiated by the substitution $y := t$. This is possible because y satisfies the eigenvariable condition.

In order to illustrate the method, we treat the first equivalence in Figure 4. In order to prove its correctness, one has to prove that for every interpretation $I = (D, \mathbf{f}, \mathbf{t}, \sqsubseteq, [\,])$,

$$I(A \wedge B) = I(\ \langle A \rangle B\ ) \vee (\ \langle B \rangle A\ ).$$

This can be done by assigning A and B values from $\{\mathbf{f}, \mathbf{t}, \mathbf{e}_1, \mathbf{e}_2\}$ and evaluating both sides of the equation, using Definition 4. The values $\mathbf{f}, \mathbf{t}$ are the truth constants, while $\mathbf{e}_1, \mathbf{e}_2$ are generic, non-Boolean elements of D. The program represents all elements of D that are not truth-values by objects of form $\sqcap S$, where S is a non-empty subset of $\{\mathbf{e}_1, \mathbf{e}_2\}$. In general, if the equivalence has n variables, one needs n constants. Since $\sqcap$ is idempotent and commutative, the set S can always be normalized by removing repeated elements and sorting the constants in an arbitrary way.

The example contains two variables, which take values from $\{\mathbf{f}, \mathbf{t}, \mathbf{e}_1, \mathbf{e}_2\}$, so that there are 4^2 possibilities to check. The following table contains a few of the possibilities:

A	B	$A \wedge B$	$\langle A \rangle B$	$\langle B \rangle A$	$(\langle A \rangle B) \vee (\langle B \rangle A)$
$\mathbf{f}$	$\mathbf{f}$	$\mathbf{f}$	$\mathbf{f}$	$\mathbf{f}$	$\mathbf{f}$
$\mathbf{f}$	$\sqcap\{\mathbf{e}_1\}$	$\sqcap\{\mathbf{e}_1\}$	$\mathbf{f}$	$\sqcap\{\mathbf{e}_1\}$	$\sqcap\{\mathbf{e}_1\}$
$\sqcap\{\mathbf{e}_1\}$	$\mathbf{t}$	$\sqcap\{\mathbf{e}_1\}$	$\sqcap\{\mathbf{e}_1\}$	$\sqcap\{\mathbf{e}_1\}$	$\sqcap\{\mathbf{e}_1\}$
$\sqcap\{\mathbf{e}_1\}$	$\sqcap\{\mathbf{e}_2\}$	$\sqcap\{\mathbf{e}_1, \mathbf{e}_2\}$	$\sqcap\{\mathbf{e}_1, \mathbf{e}_2\}$	$\sqcap\{\mathbf{e}_1, \mathbf{e}_2\}$	$\sqcap\{\mathbf{e}_1, \mathbf{e}_2\}$
$\mathbf{t}$	$\mathbf{t}$	$\mathbf{t}$	$\mathbf{t}$	$\mathbf{t}$	$\mathbf{t}$

The correctness of $I(\ (\langle A \rangle B) \vee (\langle B \rangle A)\ ) = I(A \wedge B)$ can be checked by comparing the corresponding table entries.

Correctness of propositional sequent rules (for example Theorem 6 and Theorem 4), can be proven in a similar way. For a sequence of elements of D, one can define a predicate $\mathrm{FS}(d_1, \ldots, d_n)$ denoting that the sequent $d_1, \ldots, d_n \vdash$ fails strongly. After that, the proofs reduce to checking whether in all interpretations I, the conclusion of the sequent fails strongly iff all of its premises fail strongly. All propositional rules of Seq_{PCL} have been checked in this way. The propositional rules of Seq^2_{PCL} have also been checked in a similar way.

9 Conclusions, Future Work

We have introduced PCL, which is an extension of first-order logic supporting partial functions and explicit type reasoning. It is based on the notions of strong validity and strong unsatisfiability, which we believe capture the natural, strict behaviour of simple type systems in a first-order setting. Strong validity makes it possible to express complex typing conditions in first-order logic, but at the same type keep strictness.

We introduced two sequent calculi for PCL. The first calculus Seq_{PCL} closely follows the multi-valued semantics of PCL, but it seems not suitable for practical usage, because of its unpleasant proof theoretic properties. The second calculus Seq^2_{PCL} seems to have all features that are required for practical use. Proving and type checking are separated, proofs can be easily presented, and it allows cut elimination.

Handling of partial functions was one of the motivations that were mentioned in [11] for introducing geometric resolution. The current paper is a result of this. Our aim is to extend geometric resolution (and its implementation in Geo [9]), so that it can handle PCL.

This research was funded by the ‘Programme for the Advancement of Computer Science’ of the city of Wrocław.

References

1. Berezin, S., Barrett, C., Shikanian, I., Chechik, M., Gurfinkel, A., Dill, D.: A practical approach to partial functions in CVC Lite. In: Selected Papers from the Workshop on Disproving and the Second International Workshop on Pragmatics of Decision Procedures (PDPAR 04), *Electronic Notes in Theoretical Computer Science*, vol. 125, pp. 13–23. Elsevier (2005)
2. Darvas, Á., Mehta, F., Rudich, A.: Efficient well-definedness checking. In: A. Armado, P. Baumgartner, G. Dowek (eds.) International Joint Conference on Automated Reasoning (IJCAR) 2008, *LNAI*, vol. 5195, pp. 100–115. Springer Verlag (2008)
3. Farmer, W.M.: Mechanizing the traditional approach to partial functions. In: W. Farmer, M. Kerber, M. Kohlhase (eds.) Proceedings of the Workshop on the Mechanization of Partial Functions (associated to CADE 13), pp. 27–32 (1996)
4. Farmer, W.M., Guttman, J.D.: A set theory with support for partial functions. Studia Logica **66**, 59–78 (2000)
5. Girard, J.Y., Taylor, P., Lafont, Y.: Proofs and Types. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press (1989)
6. Hähnle, R.: Many-valued logic, partiality, and abstraction in formal specification languages. Logic Journal of the IGPL **13**(4), 415–433 (2005)
7. Kerber, M., Kohlhase, M.: A mechanization of strong Kleene logic for partial functions. In: Automated Deduction - CADE 12, *LNAI*, vol. 814, pp. 371–385. Springer Verlag (1994)
8. Mehta, F.: A practical approach to partiality - a proof based approach. In: S. Liu, T. Maibaum (eds.) International Conference on Formal Engineering Methods, (ICFEM), *LNCS*, vol. 5256, pp. 238–257. Springer (2008)
9. de Nivelle, H.: Theorem prover Geo 2007F. Can be obtained from the author's homepage (2007)
10. de Nivelle, H.: Classical logic with partial functions. In: J. Giesl, R. Hähnle (eds.) International Joint Conference on Automated Reasoning (IJCAR) 2010, *LNAI*, vol. 6173, pp. 203–217. Springer Verlag (2010)
11. de Nivelle, H., Meng, J.: Geometric resolution: A proof procedure based on finite model search. In: J. Harrison, U. Furbach, N. Shankar (eds.) International Joint Conference on Automated Reasoning 2006, *Lecture Notes in Artificial Intelligence*, vol. 4130, pp. 303–317. Springer, Seattle, USA (2006)
12. Orevkov, V.: Lower bounds for increasing complexity of derivations after cut elimination. Zapiski Nauchnykh Seminarov Leningradskogo Otdeleniya Matematicheskogo Instituta Imenyi V.A. Steklova AN SSSR **88**, 137–161 (1979). English translation in Journal of Soviet Mathematics 2337–2350, 1982
13. Scott, D.: Existence and description in formal logic. In: M. Fourman (ed.) Applications of Sheaves, LNM, pp. 660–696. Springer Verlag (1977)
14. Statman, R.: Lower bounds on Herbrand's theorem. In: Proceedings of the American Mathematical Society, vol. 75-1, pp. 104–107. American Mathematical Society (1979)