Theorem Proving for Classical Logic with Partial Functions by Reduction to Kleene Logic

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

May 19, 2014

Abstract

Partial functions are abundant in mathematics and program specifications. Unfortunately, their importance has been mostly ignored in automated theorem proving. In this paper, we develop a theorem proving strategy for Partial Classical Logic (PCL). Proof search takes place in Kleene Logic. We show that PCL theories can be translated into equivalent sets of formulas in Kleene logic. For proof search we use a three-valued adaptation of geometric resolution. We prove that the calculus is sound and complete.

Keywords: Automated Theorem Proving, Partial Functions, Three-Valued Logic, Sequent Calculus, Finite Model Search.

1 Introduction and Motivation

Partial Classical Logic (PCL) was introduced in [8] with the goal of being able to deal with rich type systems and partial functions. Since the preconditions of partial functions can be complicated, as complicated as the proof itself, the preconditions have to be expressed in the same language as the reasoning problem.

A second design aim of PCL is that there should be no distinction between preconditions and types. For example, if P is a pointer type and p has type P, then retrieving $\star p$ is possible when p is not the null pointer. One can either treat the type of N non-null pointers as a subtype of P, and treat $\star$ as a total, unary function on N, or treat $\star$ as a partial function on P that is defined when $p \neq 0$. The resulting semantics should be the same.

This is in contrast with existing formalisms for partial functions that start with an underlying, simple type theory, and add partial functions by allowing functions to have preconditions expressed by logical formulas. This approach is followed in [12], [13]. In PCL, both the underlying type theory and the preconditions, are expressed in the same way by logical formulas. Figure 1 gives

Figure 1: A Simply Typed Theory

N: Type	$\forall x\ \#N(x)$
0: N	$N(0)$
S: $N \to N$	$\forall x\ N(x) \to N(S(x))$
$+$: $N \times N \to N$	$\forall xy\ N(x) \wedge N(y) \to N(x+y)$
$\equiv$: $N \times N \to$ Prop	$\forall xy\ N(x) \wedge N(y) \to \#(x \equiv y)$
$\forall x$: $N\ x + 0 \equiv x$	$\forall x\ [\ N(x)\ ]\ x + 0 \equiv x$
$\forall x, y$: $N\ x + S(y) \equiv S(x+y)$	$\forall xy\ [\ N(x) \wedge N(y)\ ]\ x + S(y) \equiv S(x+y)$

Figure 2: Specification of Subtraction

$\leq$: $N \to N \to$ Prop	$\forall xy\ N(x) \wedge N(y) \to \#(x \leq y)$
$\forall x$: $N\ 0 \leq x$	$\forall x\ [\ N(x)\ ]\ 0 \leq x$
$\forall x, y$: $N\ x \leq y \leftrightarrow S(x) \leq S(y)$	$\forall x, y\ [\ N(x) \wedge N(y)\ ]\ x \leq y \leftrightarrow S(x) \leq S(y)$
$-$: $N \to N \to N$	$\forall xy\ [\ N(x) \wedge N(y)\ ]\ (y \leq x) \to N(x-y)$
$\forall x$: $N\ x - 0 \equiv x$	$\forall x\ [\ N(x)\ ]\ x - 0 \equiv 0$
$\forall x, y$: $N\ S(x) - S(y) \equiv x - y$	$\forall xy\ [\ N(x) \wedge N(y)\ ]\ [\ y \leq x\ ]\ S(x) - S(y) \equiv x - y$

an example of a simply typed theory, expressed in PCL. Formulas in the same row have the same meaning. We assume that $\equiv$ is properly axiomatized. Type conditions connected to universal quantifiers are relativized by implications. In order to obtain type strictness, a lazy implication operator $[A]B$ is used. There also exists a lazy conjunction operator $\langle A\rangle B$ that is used in combination with existential quantifiers.

When the formulas involved are well-typed, $[A]B$ has the same semantics as $A \to B$ in untyped, classical logic. In order for $[A]B$ to be well-typed, the first argument A must be well-typed. When A is true, the second argument B has to be well-typed. The last formula in the example can be replaced $\forall xy\ N(x) \wedge N(y) \to x + S(y) \approx S(x+y)$ without changing the meaning, but only because it is well-typed. If the formula would not have been well-typed, we would have replaced a meaningless formula by a formula that probably means something unwanted. This is the general reason why typing exists: Type systems and the possibility to enforce preconditions of partial functions are features that make it easier to write down correct formulas because they allow separate checking. Typing does not increase the strength of what can be meaningfully expressed in the logic, because typing conditions can always be relativized. The situation is similar to type systems in programming languages: Type systems do not make more programs possible, but they make it easier to write correct programs because they provide automatic checking. A programmer who never makes mistakes does not need type checking.

In Example 2, a predicate $\leq$ and a partial function $(-)$ were added. The

Figure 3: Formalization of List

(1) $\forall d\ \#D(x)$
(2) $\forall l\ \#L(l)$
(3) $\forall d_1 d_2\ D(d_1) \wedge D(d_2) \rightarrow \#(d_1 \equiv d_2)$
(4) $\forall l_1 l_2\ L(l_1) \wedge L(l_2) \rightarrow \#(l_1 \equiv l_2)$
(5) $\forall l\ L(l) \rightarrow \#L_n(l) \wedge \#L_c(l)$
(6) $\forall l\ [\ L(l)\ ]\ (\ L_n(l) \vee L_c(l)\ ) \wedge (\ \neg L_n(l) \vee \neg L_c(l)\ )$
(7) $L(\mathrm{nil}) \wedge \forall d\, l\ D(d) \wedge L(l) \rightarrow L(\mathrm{cons}(d, l))$
(8) $\forall l\ [\ L(l)\ ]\ L_n(l) \leftrightarrow l \equiv \mathrm{nil}$
(9) $\forall l\ [\ L(l)\ ]\ L_c(l) \leftrightarrow \exists d\, l_1\ \langle D(d) \wedge L(l')\ \rangle\ l \equiv \mathrm{cons}(d, l')$
(10) $\forall l\ [\ L(l)\ ]\ L_c(l) \rightarrow D(\mathrm{first}(l)) \wedge L(\mathrm{rest}(l))$
(11) $\forall d\, l\ [\ D(d)\ ]\ [\ L(l)\ ]\ \mathrm{first}(\mathrm{cons}(d, l)) \equiv d \wedge \mathrm{rest}(\mathrm{cons}(d, l)) \equiv l$

predicate $\leq$ is defined everywhere on N. The function $(-)$ is defined only when the first argument is not smaller than the second. In the row where $-$ is declared, the left and the right column in Figure 2 are not equivalent anymore. In the right column, the precondition of $(-)$ can be expressed. It follows from the axiomatization of $\leq$, that for every occurrene of $t_1 - t_2$, it is possible to prove $t_2 \leq t_1$. If one would want to express the precondition of $(-)$in the left column, one would have to extend the language with a formalism for expressing the precondition. This causes the type system to be different from the language in which preconditions are expressed. This was done e.g in [12], [13], [14], or [7]. Alternatively, one can relativize the precondition using $\rightarrow$, but if one does so, one looses the possibility to have it typechecked. In the right column, type conditions are preconditions can be stated together, and they will be enforced.

Figure 3 contains a formalization of a list datatype. The first two formulas introduce a datatype D and a type of lists L. Formulas (5) and (6) state that L can be partitioned into L_n (nil), and L_c (lists constructed by cons.) Formula (10) defines the functions first and rest as partial functions that are defined on L_c. The pattern of Example 3 can be applied to all inductively defined types. Every inductive types has subtypes that correspond to the constructors and partial selector functions that are defined only for terms that are constructed by a given constructor. Using Example 3, one can check type correctness of the formula $A =$

$$\forall l\, m\ [\ L(m) \wedge L(n)\ ]\ \begin{array}{l} [\ m \equiv \mathrm{nil}\ ]\ \mathrm{append}(L, M) \equiv \mathrm{nil} \\ [\ m \not\equiv \mathrm{nil}\ ]\ \mathrm{append}(L, M) \equiv \mathrm{cons}(\mathrm{first}(l), \mathrm{append}(\mathrm{rest}(l), m)) \end{array}$$

which is the translation of the program

define append(l, m)
if $l \equiv$ nil **then return** m
else return cons(first(l), append$(\mathrm{rest}(l), m)$).

Using the specification in Example 3, it is possible to prove $\#A$, where A is

the formula above. In order to prove type correctness of the second part of A, one has to prove that $m \not\equiv \text{nil}$ implies $L_c(m)$, in order to be able to apply first and rest on L. The formula A can be translated into classical logic, by replacing [ ] by $\rightarrow$. This formula means the same, only because it is well-typed, but it would mean something unwanted if A would contain type incorrectness or ignored preconditions. Dependent on the mistake made, this may result in a specification that is stronger than intended, and which may be hard to detect.

2 Introduction to PCL

In this section, we will introduce the semantics of PCL, and provide a sequent calculus. We do this not only to make the paper self-contained, but also because there have been changes in the semantics and the sequent calculus for PCL since [8]. The changes were motivated by experience with the use of PCL in interactive systems, and by insights that were obtained when writing the current paper. The transformation to Kleene logic that is used in this paper for automated theorem proving, and which is based on *widening* (Definition 2.15), can also be used for defining a sequent calculus. The resulting sequent calculus has simpler rules than the calculus that was used in [8]. Since it is closely related to the theorem proving method, it is suitable to introduce it together.

In addition to the changes in the sequent calculus, there has been changes in the semantics as well: In earlier versions of PCL, we believed that all type information should be expressed by user defined predicates, and as a consequence, the Boolean truth values could be mixed with the domains of discourse introduced by the user. It was left to the user to define predicates that separate the different types. There was no syntactic distinction between atoms and terms, and Prop() was viewed as a predefined function symbol, that could be applied on every expression. This approach led to a couple of artefacts, that are hard to handle by deduction rules, and which at the same time are not meaningful. For example, it was possible to write the formula $\exists x_1 x_2\ \text{Prop}(x_1) \wedge \text{Prop}(x_2) \rightarrow x_1 \not\approx x_2$, which is valid, because there are two distinct Booleans in every interpretation.

Our current view is that such formulas should not be allowed. If the user wants to use a Boolean data type, this datatype should be defined by the user, and it should be separate from the truth values used by the logic. Hence, we will distinguish atoms and terms, and Prop will become a unary logical operator, similar to negation. A consequence of this modification is that we can reduce the number of error values to one. In [8], it was necessary to allow multiple error values, in order to avoid that $\neg\text{Prop}(a) \wedge \neg\text{Prop}(b) \rightarrow a \approx b$ is a tautology. But since this formula cannot be stated anymore, there is no need to worry about its truth. To reflect the fact that Prop() is not a function but a logical operator, we will write $\#A$ instead of $\text{Prop}(A)$.

The calculus that we will define in this paper is called $\text{Seq}^{\preceq}_{PCL}$. The calculus that was used in [8] was called Seq^{2}_{PCL}. The main difference is as follows: Since PCL has quite a few logical operators, both calculi use reduction rules to replace most of the operators by a small set of operators. In this way, reasoning rules

need to be defined only for some primitive operators. A similar approach is often used in classical logic. If one defines a rewrite rule that replaces $A \to B$ by $\neg A \vee B$, then one does not need reasoning rules for $\to$ anymore. The rewrite rules of Seq^2_{PCL} were truth-value preserving. During the development of theorem proving strategies, it turned out that one can obtain a better calculus if one uses *widening* (defined in Definition 2.15) instead of truth equivalence. The resulting calculus has fewer reasoning rules, and the reasoning rules are simpler. Since we use notation $\preceq$ for the widening relation, the new calculus is $\text{Seq}^{\preceq}_{PCL}$.

We will first define the formulas of PCL. After that, we define the semantics, and the notion of context.

Definition 2.1. *We first define* terms*: If f is a function symbol with arity $n \geq 0$, and $t_1, \ldots, t_n$ are terms, then $f(t_1, \ldots, t_n)$ is a term as well.*
Using terms, we define the set of formulas as follows:

- $\bot, E$ *and* $\top$ *are formulas.*
- *If* t_1 *and* t_2 *are terms, then* $t_1 \approx t_2$ *is a formula.*
- *If* p *is a predicate symbol with arity* $n \geq 0$*, and* $t_1, \ldots, t_n$ *are terms, then* $p(t_1, \ldots, t_n)$ *is a formula.*
- *If* F *is a formula, then* $\neg F$ *and* $\#F$ *are formulas.*
- *If* F *and* G *are formulas, then* $F \vee G,\ F \wedge G,\ F \to G$*, and* $F \leftrightarrow G$ *are formulas.*
- *If* F *and* G *are formulas, then* $\langle F \rangle G$ *and* $[F]G$ *are formulas.*
- *If* F *is a formula, and* x *is a variable, then* $\forall x\ F$ *and* $\exists x\ F$ *are formulas.*

We use $\approx$ for equality in formulas, in order to distinguish it from meta equality, so that we can write '$I(t_1 \approx t_2) = \mathbf{t}$'. We define three-valued interpretations:

Definition 2.2. *An* interpretation *is an object of form* $I = (D, \mathbf{f}, \mathbf{e}, \mathbf{t}, [\ ])$*, where* $\mathbf{f}, \mathbf{e}, \mathbf{t}$ *are distinct objects. The function* $[\ ]$ *interprets function symbols as functions from* D^n *to* D *and predicate symbols as functions from* D^n *to* $\{\mathbf{f}, \mathbf{e}, \mathbf{t}\}$*, in accordance with the arity of the symbol.*

Definition 2.3. *Let* $I = (D, \mathbf{f}, \mathbf{e}, \mathbf{t}, [\ ])$ *be an interpretation. Starting with* $[\ ]$*, we define the interpretation function* $I()$*, that interprets all terms and formulas.*

- *If* f *is a function symbol of arity* n*, and* $t_1, \ldots, t_n$ *are terms, then* $I(\ f(t_1, \ldots, t_n)\ ) = [f](I(t_1), \ldots, I(t_n))$.
- *If* p *is a predicate symbol of arity* n*, and* $t_1, \ldots, t_n$ *are terms, then* $I(\ p(t_1, \ldots, t_n)\ ) = [p](I(t_1), \ldots, I(t_n))$.
- $I(\bot) = \mathbf{f},\ I(E) = \mathbf{e}$*, and* $I(\top) = \mathbf{t}$.

Figure 4: Semantics of Logical Operators

- The semantics of $\neg$ and $\#$ is defined by the following truth tables:

$$\neg: \begin{array}{|c|}\hline \mathbf{t} \\ \hline \mathbf{e} \\ \hline \mathbf{f} \\ \hline \end{array} \qquad \# : \begin{array}{|c|}\hline \mathbf{t} \\ \hline \mathbf{f} \\ \hline \mathbf{t} \\ \hline \end{array}$$

- The semantics of the strict operators $\vee$, $\wedge, \rightarrow$ and $\leftrightarrow$ is defined by the following truth tables:

$$\vee: \begin{array}{|c|c|c|}\hline \mathbf{f} & \mathbf{e} & \mathbf{t} \\ \hline \mathbf{e} & \mathbf{e} & \mathbf{e} \\ \hline \mathbf{t} & \mathbf{e} & \mathbf{t} \\ \hline \end{array} \quad \wedge: \begin{array}{|c|c|c|}\hline \mathbf{f} & \mathbf{e} & \mathbf{f} \\ \hline \mathbf{e} & \mathbf{e} & \mathbf{e} \\ \hline \mathbf{f} & \mathbf{e} & \mathbf{t} \\ \hline \end{array} \quad \leftrightarrow: \begin{array}{|c|c|c|}\hline \mathbf{t} & \mathbf{e} & \mathbf{f} \\ \hline \mathbf{e} & \mathbf{e} & \mathbf{e} \\ \hline \mathbf{f} & \mathbf{e} & \mathbf{t} \\ \hline \end{array} \quad \rightarrow: \begin{array}{|c|c|c|}\hline \mathbf{t} & \mathbf{e} & \mathbf{t} \\ \hline \mathbf{e} & \mathbf{e} & \mathbf{e} \\ \hline \mathbf{f} & \mathbf{e} & \mathbf{t} \\ \hline \end{array}$$

- The semantics of the weak operators $\langle\ \rangle$ and $[\ ]$ is defined by the following truth tables:

$$\langle\ \rangle: \begin{array}{|c|c|c|}\hline \mathbf{f} & \mathbf{f} & \mathbf{f} \\ \hline \mathbf{e} & \mathbf{e} & \mathbf{e} \\ \hline \mathbf{f} & \mathbf{e} & \mathbf{t} \\ \hline \end{array} \qquad [\ ]: \begin{array}{|c|c|c|}\hline \mathbf{t} & \mathbf{t} & \mathbf{t} \\ \hline \mathbf{e} & \mathbf{e} & \mathbf{e} \\ \hline \mathbf{f} & \mathbf{e} & \mathbf{t} \\ \hline \end{array}$$

- The semantics of the quantifiers is defined by the following preferences:

$$\forall : \mathbf{e} > \mathbf{f} > \mathbf{t}, \qquad \exists : \mathbf{e} > \mathbf{t} > \mathbf{f}.$$

- *For a unary propositional operator $\star$, the interpretation of $I(\star A)$ is defined as the value in the corresponding table in Figure 4, using the value of $I(A)$ in the order $\mathbf{f}, \mathbf{e}, \mathbf{t}$.*

- *For a binary propositional operator $\star$, the interpretation of $I(A \star B)$ is defined as the value in the corresponding table in Figure 4. The row is determined by the value of $I(A)$ in the order $\mathbf{f}, \mathbf{e}, \mathbf{t}$, and the column is determined by the value of $I(B)$, also in the order $\mathbf{f}, \mathbf{e}, \mathbf{t}$.*

- *For a quantifier Q, the value of $I(Qx\ F)$ is obtained as follows: First define $R_F = \{I^x_d(F) \mid d \in D\}$. Then $I(Qx\ F)$ is obtained by selecting from R_F the most preferred value using the preference list for Q in Figure 4.*

In PCL, theories are constructed in *contexts*. A context is a mixture of assumptions and theorems. The order of formulas in a context is essential, because theorems must be provable from the formulas that occur before them, and types of functions and predicates have to be specified before they are used. Contexts play the role of the first component of Seq^2_{PCL}-sequents in [8].

Definition 2.4. *We call an object of form* $\|\Gamma_1, \ldots, \Gamma_m\|$, *in which* Γ_j $(m \geq 0)$ *are formulas, and in which some of the* Γ_j *are possibly marked with a* ϑ, *a* context.

Formulas that are marked with ϑ denote theorems, which means that they are provable from the formulas that occur before them. Unmarked formulas are assumptions.

Example 2.5. $\| \#A,\ \#B,\ A,\ B,\ (A \wedge B)^{\vartheta} \|$ *is an example of a context. The formula* $(A \wedge B)^{\vartheta}$ *is marked as theorem, which is correct, because it is provable from the formulas* A *and* B. *The formulas* A *and* B *can be assumed because* $\#A$ *and* $\#B$ *occur before them.*

Definition 2.6. *Let* $\|\Gamma\|$ *be a context. We say that* $\|\Gamma\|$ *is* strongly valid *if in every interpretation* I, *for which there is an* i, *s.t.* $I(\Gamma_i) \neq \mathbf{t}$, *the first such* i *satisfies the following condition:*

- Γ_i *is not marked as theorem and* $I(\Gamma_i) = \mathbf{f}$.

Strong validity captures the intuition that theorems must be provable from the formulas before them, and assumptions must be well-defined assuming the formulas before them. The following theorem, which is a generalization of Theorem 1 in [8], confirms this.

Theorem 2.7. *The context* $\|\Gamma_1, \ldots, \Gamma_m\|$ *is strongly valid iff for every interpretation* I, *for every* i $(1 \leq i \leq m)$, *the following holds:*

- *If* Γ_i *is not marked, then* $I(\Gamma_1) = \cdots = I(\Gamma_{i-1}) = \mathbf{t}$ *implies* $I(\Gamma_i) \in \{\mathbf{f}, \mathbf{t}\}$.
- *If* Γ_i *is marked, then* $I(\Gamma_1) = \cdots = I(\Gamma_{i-1}) = \mathbf{t}$ *implies* $I(\Gamma_i) = \mathbf{t}$.

Proof. First assume that $\|\Gamma_1, \ldots, \Gamma_m\|$ is strongly valid. Assume that I is an interpretation and that $I(\Gamma_1) = \cdots = I(\Gamma_{i-1}) = \mathbf{t}$. If Γ_i is not marked, then by definition of strong validity, $I(\Gamma_i) \neq \mathbf{t} \Rightarrow I(\Gamma_i) = \mathbf{f}$, so that $I(\Gamma_i) \in \{\mathbf{f}, \mathbf{t}\}$. If Γ_i is marked, then it follows from the definition of strong validity, that it cannot be the case that $I(\Gamma_i) \neq \mathbf{t}$. Hence we have shown that $I(\Gamma_i) = \mathbf{t}$.

Now suppose that the converse holds. We must show that $\|\Gamma_1, \ldots, \Gamma_m\|$ is strongly valid. Assume that there exists an i, s.t. $I(\Gamma_i) \neq \mathbf{t}$. We can assume that i is minimal. If Γ_i is marked, then we have obtained a contradiction with $I(\Gamma_1) = \cdots = I(\Gamma_{i-1}) = \mathbf{t} \Rightarrow I(\Gamma_i) = \mathbf{t}$, so that it follows that Γ_i is not marked. If Γ_i is not marked, then we have $I(\Gamma_1) = \cdots = I(\Gamma_{i-1}) \Rightarrow I(\Gamma_i) \in \{\mathbf{f}, \mathbf{t}\}$, so that we have shown that $I(\Gamma_i) = \mathbf{f}$. □

Example 2.8. *The context* $\| A,\ B,\ (A \wedge B)^{\vartheta} \|$ *is not strongly valid, because Definition 2.6 is not met by* I *with* $I(A) = \mathbf{e}$, $I(B) = \mathbf{t}$. *The context* $\| \#A,\ A,\ B,\ (A \wedge B)^{\vartheta} \|$ *is still not strongly valid, because it is possible that* $I(B) = \mathbf{e}$, *which would make* B *with* $I(B) \neq \mathbf{f}$ *the first formula with* $I(B) \neq \mathbf{t}$.

In order to make the context strongly valid, we also have to add $\#B$, *so that we get* $\| \#A,\ \#B,\ A,\ B,\ (A \wedge B)^{\vartheta} \|$.

Example 2.9. *The context* $\| \ G(s),\ \forall x\ G(x) \to M(x),\ M^{\vartheta}(s) \ \|$ *is not strongly valid, despite the fact that* $G(s),\ \forall x\ G(x) \to M(x)$ *implies* $M(s)$.

In order to make the context strongly valid, one has to make sure that predicates G *and* M *(Greek and Mortal) are always well-defined:*

$$\| \ \forall x\ \#G(x),\ \forall x\ \#M(x),\ G(s),\ \forall x\ G(x) \to M(x),\ M^{\vartheta}(s) \ \|.$$

The predicates G *and* M *are currently too general. In order to be more realistic, they can be restricted to be subpredicates of a predicate* H *(Human):*

$$\left\| \begin{array}{l} \forall x\ \#H(x) \\ \forall x\ H(x) \to \#G(x) \\ \forall x\ H(x) \to \#M(x) \\ \langle H(s) \rangle\ G(s) \\ \forall x\ [H(x)]\ G(x) \to M(x) \\ M^{\vartheta}(s) \end{array} \right\|$$

If the formula $\langle H(s) \rangle\ G(s)$ *would be replaced by* $H(s) \wedge G(s)$*, the resulting context would be not strongly valid anymore, because one can have* $I(H(s)) = \mathbf{f},\ I(G(s)) = \mathbf{e}$*, while making all the formulas before it true.*

Similarly, replacing $\forall x\ [H(x)]\ G(x) \to M(x)$ *by* $\forall x\ H(x) \wedge G(x) \to M(x)$ *would make the context not strongly valid anymore, because there could exist a term* t *with* $I(H(t)) = \mathbf{f},\ I(G(t)) = \mathbf{e}$.

Our goal in the current section is to describe the sequent calculus $\mathrm{Seq}^{\preceq}_{PCL}$, which is an improvement of Seq^{2}_{PCL}. Calculus Seq^{2}_{PCL} is based on Theorem 2.7, which makes it possible to replace a context by a set of standard implications, which can be proven by traditional means. The calculus $\mathrm{Seq}^{\preceq}_{PCL}$ is based on a stronger equivalence, which we will prove shortly.

Definition 2.10. *A* sequent *is an object of form* $S \vdash \bot$*, in which* S *is a set of formulas.*

Definition 2.11. *We call a set of formulas* S unsatisfiable *if for every interpretation* I*, there is a formula* $A \in S$*, s.t.* $I(A) \neq \mathbf{t}$*. We say that a set of sequents* $\{S_1 \vdash \bot,\ \ldots,\ S_n \vdash \bot\}$ represents *a property* P *if* $P \Leftrightarrow$ *all* S_i *are unsatisfiable.*

Definition 2.12. *Let* $\|\Gamma\| = \|\Gamma_1, \ldots, \Gamma_m\|$ *be a context. The* sequent expansion $\mathrm{Seq}(\ \|\Gamma\|\ )$ *of* $\|\Gamma\|$ *is defined as* $\bigcup_{1 \leq i \leq m} E(\ \|\Gamma\|,\ i) \cup \bigcup_{1 \leq i \leq m} E_{\vartheta}(\ \|\Gamma\|,\ i)$*, where*

- $E(\ \|\Gamma\|,\ i) = \{\ \{\Gamma_1, \ldots, \Gamma_{i-1}, \neg(\#\Gamma_i)\} \vdash \bot\ \}$.
- $E_{\vartheta}(\ \|\Gamma\|,\ i) = \{\ \{\Gamma_1, \ldots, \Gamma_{i-1}, \neg\Gamma_i\} \vdash \bot\ \}$ *if* Γ_i *is marked as theorem, and* $\{\ \}$ *otherwise.*

It is clear that Definition 2.12 is inspired by Theorem 2.7. For unmarked Γ_i, the set $E(\|\Gamma\|, i)$ represents the property $I(\Gamma_1) = \cdots = I(\Gamma_{i-1}) = \mathbf{t} \Rightarrow I(\Gamma_i) \in \{\mathbf{f}, \mathbf{t}\}$. For marked Γ_i, the set $E(\|\Gamma\|, i) \cup E_{\vartheta}(\|\Gamma\|, i)$ represents the property $I(\Gamma_1) = \cdots = I(\Gamma_{i-1}) = \mathbf{t} \Rightarrow I(\Gamma_i) = \mathbf{t}$. It follows immediately from Theorem 2.7 that $\mathrm{Seq}(\|\Gamma\|)$ represents the property '$\|\Gamma\|$ is strongly valid'. The

sequent calculus Seq^2_{PCL} used this equivalence. It has a rule called 'Extension', by which, in a somewhat untransparant fashion, one could construct the expansion. The reasoning rules

$$\frac{S_1 \vdash \bot \ \cdots \ S_n \vdash \bot}{S \vdash \bot}$$

of Seq^2_{PCL} preserve unsatisfiability, which means that for every rule, S is unsatisfiable iff all S_i are unsatisfiable. The further design of Seq^2_{PCL} is standard. It was presented as a sequent calculus, but it can be presented as a tableau calculus as well.

We will now introduce a new calculus called $\mathrm{Seq}^{\preceq}_{PCL}$, which is based on the observation that $\mathrm{Seq}(\|\Gamma\|)$ represents validity of $\|\Gamma\|$ in a strong fashion. We will first define strong representation, then prove that it holds, and after that, introduce the calculus. We will also explain why $\mathrm{Seq}^{\preceq}_{PCL}$ is better than Seq^2_{PCL}.

Definition 2.13. *We say that a set of formulas S is* strongly unsatisfiable *if for every interpretation I, there is a formula $A \in S$, s.t. $I(A) = \mathbf{f}$.*

We say that a set of sequents $S_1 \vdash \bot, \ldots, S_n \vdash \bot$ strongly represents a property P *iff*

1. *P is true implies that all of the S_i are strongly unsatisfiable.*
2. *P is false implies that (at least) one of the S_i is satisfiable.*

Theorem 2.14. *For every context Γ, the sequent expansion $\mathrm{Seq}(\|\Gamma\|)$ strongly represents the property that $\|\Gamma\|$ is strongly valid.*

Proof. We first prove (1) of Definition 2.13. Assume that $\|\Gamma\|$ is strongly valid. Write $\|\Gamma\| = \|\Gamma_1, \ldots, \Gamma_m\|$. Let I be an arbitrary interpretation. If there exists an $i \in \{1, \ldots, m\}$, s.t. $I(\Gamma_i) \neq \mathbf{t}$, then let i be the smallest such number. Otherwise, leave i undefined.

Let $S \vdash \bot$ be a sequent that occurs in $\mathrm{Seq}(\|\Gamma\|)$. Let λ be the level on which $S \vdash \bot$ was introduced. We have to show that there exists an $A \in S$, s.t. $I(A) = \mathbf{f}$.

- If i is undefined or $\lambda < i$, then we have $I(\Gamma_\lambda) = I(\#\Gamma_\lambda) = \mathbf{t}$, so that $I(\neg\Gamma_\lambda) = I(\neg\#\Gamma_\lambda) = \mathbf{f}$. If $(S \vdash \bot) \in E(\ \|\Gamma\|,\ \lambda)$, then $\neg\#\Gamma_\lambda \in S$, so that one can take $A = \neg\#\Gamma_\lambda$. If $(S \vdash \bot) \in E_\vartheta(\ \|\Gamma\|,\ \lambda)$, then $\neg\Gamma_\lambda \in S$, so that one can take $A = \neg\Gamma_\lambda$.

- If i is defined and $\lambda = i$, then it follows from the definition of strong validity, that Γ_i is not marked. This implies that $E_\vartheta(\ \|\Gamma\|,\ i) = \emptyset$, so that $(S \vdash \bot) \in E(\ \|\Gamma\|,\ i)$. By strong validity of $\|\Gamma\|$, we know that $I(\Gamma_i) = \mathbf{f}$, so that $I(\neg\#\Gamma_i) = \mathbf{f}$ as well. Since $\neg\#\Gamma_i \in S$, we can take $A = \neg\#\Gamma_i$.

- If i is defined and $\lambda > i$, then it follows from strong validity of $\|\Gamma\|$, that $I(\Gamma_i) = \mathbf{f}$. We have either $(S \vdash \bot)$ in $E(\ \|\Gamma\|,\ \lambda)$, or $(S \vdash \bot)$ in $E_\vartheta(\ \|\Gamma\|,\ \lambda)$. In both cases, we have $\Gamma_i \in S$, so that we can take $A = \Gamma_i$.

Next we prove (2) of Definition 2.13. Assume that $\|\Gamma\|$ is not strongly valid. This means that there exist an interpretation I and an i, s.t. $I(\Gamma_i) \neq \mathbf{t}$, where the first such i has either $I(\Gamma_i) \neq \mathbf{f}$, or Γ_i is marked. This can be reformulated as either $I(\Gamma_i) = \mathbf{e}$, or $I(\Gamma_I) = \mathbf{f}$ and Γ_i is marked. In both cases, $\lambda < i \Rightarrow I(\Gamma_\lambda) = \mathbf{t}$.

- If $I(\Gamma_i) = \mathbf{e}$, then $I(\neg\#\Gamma_i) = \mathbf{t}$. It follows that all formulas in the premise of $\{\Gamma_1, \ldots, \Gamma_{i-1}, \neg\#\Gamma_i\} \vdash \bot$ are true in I. Since this sequent is included in $E(\ \|\Gamma\|,\ i)$, the proof is complete for this case.
- If Γ_i is marked and $I(\Gamma_i) = \mathbf{f}$, then $I(\neg\Gamma_i) = \mathbf{t}$. It follows that all formulas in the premise of $\{\Gamma_1, \ldots, \Gamma_{i-1}, \neg\Gamma_i\} \vdash \bot$ are true in I. Since this sequent is included in $E_\vartheta(\ \|\Gamma\|,\ i)$, the proof is complete.

□

The advantage of strong representation is that it makes it possible to ignore error values during proof search. Either, it is possible to make all formulas in all sequents true, or there always is a false formula in some sequent. A calculus that is based on standard unsatisfiability has to preserve error values during proof search, since otherwise it might replace an unsatisfiable sequent by a satisfiable sequent. A calculus based on strong representation can ignore error values. We prove that error values can be ignored:

Definition 2.15. *We define the* widening relation $\preceq$ *as follows:* $A \preceq B$ *if in every interpretation* I*, we have*

$$\begin{array}{rcl} I(A) = \mathbf{f} & \Rightarrow & I(B) = \mathbf{f}, \\ I(A) = \mathbf{t} & \Rightarrow & I(B) = \mathbf{t}. \end{array}$$

We write $A \equiv B$ *if* $A \preceq B$ *and* $B \preceq A$*. It can be easily checked that* $A \equiv B$ *iff* $I(A) = I(B)$ *in every interpretation* I*. In this case, we call the formulas* equivalent*.*

Theorem 2.16. *Let* $\{S_1 \vdash \bot,\ \ldots,\ S_n \vdash \bot\}$ *be a set of sequents that strongly represents a property* P*. Let* A *and* B *be two formulas with* $A \preceq B$*.*

Let $\{S_1' \vdash \bot,\ \ldots,\ S_n' \vdash \bot\}$ *be obtained from* $\{S_1 \vdash \bot,\ \ldots,\ S_n \vdash \bot\}$ *by possibly replacing some occurrences of* A *by* B*.*

Then $\{S_1' \vdash \bot,\ \ldots,\ S_n' \vdash \bot\}$ *strongly represents property* P *as well.*

Proof. First assume that P is true. In that case all S_i are strongly unsatisfiable. Let I be an interpretation. Let S_i be one of the $S_1, \ldots, S_n$. By Definition 2.13, S_i contains a formula C with $I(C) = \mathbf{f}$. If still $C \in S_i'$, then we are done. Otherwise, it must be the case that $C = A$ and $B \in S_i'$. Since $A \preceq B$, it follows that $I(B) = \mathbf{f}$.

Next assume that P is false. This implies that one of the S_i is satisfiable, which means that there is an interpretation I, in which all formulas of S_i are true. We show that all formulas in S_i' are true in I as well. Let C be a formula in S_i'. If $C \in S_i$, then we are done. Otherwise, we have $C = B$ and $A \in S_i$. Since $I(A) = \mathbf{t}$ and $A \preceq B$, we have $I(B) = \mathbf{t}$, so that the proof is complete. □

Figure 5: Semantics of Kleene Operators

- The semantics of the binary operators $\otimes$ and $\oplus$ is defined by the following truth tables:

$\otimes$:

f	**f**	**f**
f	**e**	**e**
f	**e**	**t**

$\oplus$:

f	**e**	**t**
e	**e**	**t**
t	**t**	**t**

- The semantics of the quantifiers Π and Σ is defined by the following preferences:

$$\Pi : \mathbf{f} > \mathbf{e} > \mathbf{t}, \qquad \Sigma : \mathbf{t} > \mathbf{e} > \mathbf{f}.$$

The calculus $\mathrm{Seq}^{\preceq}_{PCL}$ is based on the fact that the natural sequent rules for Kleene logic preserve strong representation, in combination with the fact that all PCL operators can be widened into their corresponding Kleene operators by simple replacement rules. The result is a calculus that does not have many reasoning rules, and that is easy to use. We will use the notation $\otimes$ for Kleene conjunction, $\oplus$ for Kleene disjunction, Π for Kleene universal quantification, and Σ for Kleene existential quantification.

Definition 2.17. *We extend the set of formulas, defined in Definition 2.1, with two binary operators and two quantifiers as follows:*

- *If F and G are formulas, then $F \otimes G$ and $F \oplus G$ are formulas as well.*
- *If F is a formula and x is a variable, then $\Pi x\ F$ and $\Sigma x\ F$ are formulas.*

We extend the interpretation of formulas, defined in Definition 2.3, by the tables in Figure 5.

Theorem 2.18. *For every rule of Figure 6, that is written as $A \preceq B$, it is indeed the case that $A \preceq B$. For every rule that is written as $A \equiv B$, it is indeed the case that $A \equiv B$.*

We can now define the deduction rules of $\mathrm{Seq}^{\preceq}_{PCL}$. The rules are listed in Figure 7. We will prove that backward application of a deduction rule on a sequent preserves strong representation. In order to do this, we first prove the following property:

Theorem 2.19. *Let $\frac{S \cup R_1 \vdash \bot,\ ...,\ S \cup R_p \vdash \bot}{S \cup R \vdash \bot}$ be one of the deduction rules of Figure 7.*

1. *If $S \cup R$ is satisfiable, then there is an i $(1 \leq i \leq p)$, s.t. $S \cup R_i$ is satisfiable.*
2. *If $S \cup R$ is strongly unsatisfiable, then for every i $(1 \leq i \leq p)$, $S \cup R_i$ is strongly unsatisfiable.*

Figure 6: Kleening Rules of $\mathrm{Seq}^{\preceq}_{\overline{PCL}}$

Rules for PCL operators (left) and NNF rules (right):

$$
\begin{array}{lcl}
A \wedge B & \preceq & A \otimes B \\
A \vee B & \preceq & A \oplus B \\
A \to B & \preceq & \neg A \oplus B \\
A \leftrightarrow B & \preceq & (\neg A \vee B) \otimes (A \vee \neg B) \\
[A]B & \preceq & \neg A \oplus B \\
\langle A \rangle B & \preceq & A \otimes B \\
\forall x\ F[x] & \preceq & \Pi x\ F[x] \\
\exists x\ F[x] & \preceq & \Sigma x\ F[x]
\end{array}
\qquad
\begin{array}{lcl}
\neg\bot & \equiv & \top \\
\neg E & \equiv & E \\
\neg\top & \equiv & \bot \\
\neg\neg A & \equiv & A \\
\neg(A \oplus B) & \equiv & \neg A \otimes \neg B \\
\neg(A \otimes B) & \equiv & \neg A \oplus \neg B \\
\neg(\ \Pi x\ F[x]\ ) & \equiv & \Sigma x\ \neg F[x] \\
\neg(\ \Sigma x\ F[x]\ ) & \equiv & \Pi x\ \neg F[x]
\end{array}
$$

Rules for #:

$$
\begin{array}{lcl}
\#\top & \equiv & \top \\
\#E & \equiv & \bot \\
\#\bot & \equiv & \top \\
\#(\neg A) & \equiv & \#A \\
\#(\#A) & \equiv & \top \\
\#(\ \forall x\ F[x]\ ) & \equiv & \Pi x\ \#F[x] \\
\#(\ \exists x\ F[x]\ ) & \equiv & \Pi x\ \#F[x]
\end{array}
\qquad
\begin{array}{lcl}
\#(A \wedge B) & \equiv & \#A\ \otimes\ \#B \\
\#(A \vee B) & \equiv & \#A\ \otimes\ \#B \\
\#(A \to B) & \equiv & \#A\ \otimes\ \#B \\
\#(A \leftrightarrow B) & \equiv & \#A\ \otimes\ \#B \\
\#(\ [A]B\ ) & \equiv & \#A\ \otimes\ (\ \neg A \vee \#B\ ) \\
\#(\ \langle A \rangle B\ ) & \equiv & \#A\ \otimes\ (\ \neg A \vee \#B\ ) \\
\#(t_1 \approx t_2) & \equiv & \top
\end{array}
$$

Radicalization Rules:

$$
\begin{array}{lcl}
A & \preceq & \#A \otimes A \\
A & \preceq & \neg(\#A) \oplus A
\end{array}
$$

Figure 7: Deduction Rules of $\mathrm{Seq}^{\preceq}_{\overline{P}CL}$

Axioms

$$\frac{}{S, \bot \vdash \bot} \qquad \frac{}{S, A, \neg A \vdash \bot} \qquad \frac{}{S,\ \neg(t \approx t) \vdash \bot}$$

Equivalence

$$\frac{S,\ B \vdash \bot}{S,\ A \vdash \bot} \qquad \frac{S,\ \neg B \vdash \bot}{S,\ \neg A \vdash \bot}$$

($A \preceq B$ or $A \equiv B$ must occur in Figure 6)

Rules for $\oplus$ and $\otimes$

$$\frac{S,\ A \vdash \bot \qquad S,\ B \vdash \bot}{S,\ A \oplus B \vdash \bot} \qquad \frac{S,\ A,\ B \vdash \bot}{S,\ A \otimes B \vdash \bot}$$

Rules for Σ and Π

$$\frac{S,\ P[y] \vdash \bot}{S,\ \Sigma x\ P[x] \vdash \bot} \qquad \frac{S,\ P[t],\ \Pi x\ P[x] \vdash \bot}{S,\ \Pi x\ P[x] \vdash \bot}$$

In the Σ rule, y must be not free in S or $P[x]$. In the Π rule, t denotes an arbitrary term.

Equality

$$\frac{S,\ t_1 \approx t_2,\ A[t_2] \vdash \bot}{S,\ t_1 \approx t_2,\ A[t_1] \vdash \bot}$$

Cut

$$\frac{S,\ \neg(\#A) \vdash \bot \quad S,\ \neg A \vdash \bot \quad S,\ A \vdash \bot}{S \vdash \bot}$$

Proof. We first fix some notation that will be used throughout the proof. For a set of formulas S, we write $I(S) = \mathbf{t}$ if for every $A \in S$, $I(A) = \mathbf{t}$. We write $I(S) = \mathbf{f}$ if there is an $A \in S$, s.t. $I(A) = \mathbf{f}$. We will not give proofs for all rules, because some of them are very similar.

Axiom: Consider the first axiom, so that we have $p = 0$ and $R = \{\bot\}$. The set of formulas $S \cup \{\bot\}$ is not satisfiable, so that we may assume that there is an $1 \leq i \leq p$, s.t. $S \cup R_i$ is satisfiable, when it happens.

Since there are no i with $(1 \leq i \leq p)$, we may assume that each $S \cup R_i$ is strongly unsatisfiable.

The other two axioms are similar.

Equivalence: It can be easily checked that $A \preceq B$ implies $\neg A \preceq \neg B$. This means that in both cases, we can assume that the rule has form $\frac{S \cup \{A\}}{S \cup \{B\}}$.

If $S \cup \{A\}$ is satisfiable, then there is an interpretation I, s.t. $I(S) = I(A) = \mathbf{t}$. Since $A \preceq B$, we also have $I(B) = \mathbf{t}$, so that $S \cup \{B\}$ is satisfiable as well.

Now assume that $S \cup \{A\}$ is strongly unsatisfiable. Assume that I is an interpretation. If $I(S) = \mathbf{f}$, then we are done. Otherwise, it must be the case that $I(A) = \mathbf{f}$, by strong unsatisfiability of $S \cup \{A\}$. Since we have $A \preceq B$, we also have $I(B) = \mathbf{f}$.

$\oplus$: Assume that $S \cup \{A \oplus B\}$ is satisfiable. This means that there is an interpretation I, s.t. $I(S) = I(A \oplus B) = \mathbf{t}$. From the truth table of $\oplus$ in Figure 5, it follows that either $I(A) = \mathbf{t}$ or $I(B) = \mathbf{t}$. As a consequence, we have either $I(S \cup \{A\}) = \mathbf{t}$ or $I(S \cup \{B\}) = \mathbf{t}$.

Next assume that $S \cup \{A \oplus B\}$ is strongly unsatisfiable. For every interpretation I, we have $I(S \cup \{A \oplus B\}) = \mathbf{f}$. We show that $I(S \cup \{A\})$ is strongly unsatisfiable. Strong unsatisfiability of $I(S \cup \{B\})$ follows by symmetry. Let I be an arbitrary interpretation. If $I(S) = \mathbf{f}$, then we are done. Otherwise, it must be the case that $I(A \oplus B) = \mathbf{f}$, which implies that $I(A) = \mathbf{f}$, so that the proof is complete.

$\otimes$: Assume that $S \cup \{A \otimes B\}$ is satisfiable. There is an interpretation I, s.t. $I(S) = I(A \otimes B) = \mathbf{t}$. It follows from the truth table of $\otimes$ that $I(A) = I(B) = \mathbf{t}$, which implies that $S \cup \{A, B\}$ is satisfiable.

Next assume that $S \cup \{A \otimes B\}$ is strongly unsatisfiable. We have to show that $S \cup \{A, B\}$ is strongly unsatisfiable as well. Let I be an arbitrary interpretation. If $I(S) = \mathbf{f}$, then we are done at this point. Otherwise, it must be the case that $I(A \otimes B) = \mathbf{f}$, which implies that either $I(A) = \mathbf{f}$ or $I(B) = \mathbf{f}$. In both cases, we have $I(S \cup \{A, B\}) = \mathbf{f}$, so that the proof is complete.

Σ: Assume that $S \cup \{\Sigma x\ P[x]\}$ is satisfiable. Then there is an interpretation I, s.t. $I(S) = I(\ \Sigma x\ P[x]\ ) = \mathbf{t}$. Since y is not free in $P[x]$, we also have

$I(\ \Sigma y\ P[y]\ ) = \mathbf{t}$. Write $I = (D, \mathbf{f}, \mathbf{e}, \mathbf{t}, [\ ])$. Using the semantics of Σ, there must be a $d \in D$, s.t. $I^y_d(\ P[y]\ ) = \mathbf{t}$. Since y is not free in S, we have $I^y_d(S) = I(S) = \mathbf{t}$, so that $I^y_d(S \cup \{P[y]\}) = \mathbf{t}$.

Next we assume that $S \cup \{\Sigma x.P[x]\}$ is strongly unsatisfiable and prove that $S \cup \{P[y]\}$ is strongly unsatisfiable as well. Let I be an arbitrary interpretation of $S \cup \{P[y]\}$. If $I(S) = \mathbf{f}$, then we are done. Otherwise, it must be the case that $I(\ \Sigma x\ P[x]\ ) = \mathbf{f}$. We also have $I(\ \Sigma y\ P[y]\ ) = \mathbf{f}$, because y is not free in $P[x]$. Write $I = (D, \mathbf{f}, \mathbf{e}, \mathbf{t}, [\ ])$. By the semantics of Σ it follows that for every $d \in D$, we have $I^y_d(\ P[y]\ ) = \mathbf{f}$. Since we pick $d = I(y)$, which results in $I^y_d = I$, we obtain that $I(\ P[y]\ ) = \mathbf{f}$.

Π: Assume that $S \cup \{\Pi x\ P[x]\}$ is satisfiable. This means that there is an interpretation I, s.t. $I(S) = I(\ \Pi x\ P[x]\ ) = \mathbf{t}$. Writing $I = (D, \mathbf{f}, \mathbf{e}, \mathbf{t}, [\ ])$, we have $I^x_{I(t)}(\ P[x]\ ) = \mathbf{t}$, which implies that $I(\ P[x := t]\ ) = \mathbf{t}$, so that $I(S \cup \{\Pi x\ P[x],\ P[x := t]\}) = \mathbf{t}$.

Next assume that $S \cup \{\ \Pi x\ P[x]\ \}$ is strongly unsatisfiable. Since $S \cup \{\ \Pi x\ P[x]\ \} \subseteq S \cup \{\ \Pi x\ P[x],\ P[x := t]\ \}$, it must be the case that $S \cup \{\ \Pi x\ P[x],\ P[x := t]\ \}$ is strongly unsatisfiable as well.

Equality: Assume that $S \cup \{\ t_1 \approx t_2,\ A[t_1]\ \}$ is satisfiable. Since $I(t_1 \approx t_2) = \mathbf{t}$ implies that $I(A[t_1]) = I(A[t_2])$, it follows that $I(A[t_2]) = \mathbf{t}$, so that $S \cup \{\ t_1 \approx t_2,\ A[t_2]\ \}$ is satisfiable as well.

Now assume that $S \cup \{\ t_1 \approx t_2,\ A[t_1]\ \}$ is strongly unsatisfiable. We must show that $S \cup \{\ t_1 \approx t_2,\ A[t_2]\ \}$ is also strongly unsatisfiable. Let I be an arbitrary interpretation. We have either $I(S \cup \{t_1 \approx t_2\}) = \mathbf{f}$ or $I(A[t_1]) = \mathbf{f}$. In the first case, we are done. In the second case, we may assume that $I(S \cup \{t_1 \approx t_2\}) \neq \mathbf{f}$, which implies that $I(t_1 \approx t_2) \neq \mathbf{f}$, which in turn implies that $I(t_1 \approx t_2) = \mathbf{t}$. But then we have $I(A[t_1]) = I(A[t_2])$, so that $I(A[t_2]) = \mathbf{f}$.

Cut: Assume that S is satisfiable. This means that there exists an interpretation I with $I(S) = \mathbf{t}$. Since $I(A) \in \{\mathbf{f}, \mathbf{e}, \mathbf{t}\}$, it must be the case that $I(B) = \mathbf{t}$ for one of $\{\neg A,\ \neg(\#A),\ A\}$. This implies that $I(S \cup \{B\}) = \mathbf{t}$, so that $S \cup \{B\}$ is satisfiable.

If $I(S) = \mathbf{f}$, then $I(S \cup \{\neg A\}) = I(S \cup \{\neg(\#A)\}) = I(S \cup \{A\}) = \mathbf{f}$, because each of them contains S as a subset.

□

Theorem 2.20. *Let* $\frac{S \cup R_1 \vdash \bot,\ \ldots,\ S \cup R_p \vdash \bot}{S \cup R \vdash \bot}$ *be one of the deduction rules of Figure 7. Assume that* $\{\ S \cup R \vdash \bot,\ X_1 \vdash \bot,\ \ldots,\ X_n \vdash \bot\ \}$ *is a set of sequents that strongly represents some property P. Then*

$$\{\ S \cup R_1 \vdash \bot,\ \ldots,\ S \cup R_p \vdash \bot,\ \ X_1 \vdash \bot,\ \ldots,\ X_n \vdash \bot\ \}$$

strongly represents P as well.

Proof. First assume that P is true. By Definition 2.13, $S \cup R$ and $X_1, \ldots, X_n$ are strongly unsatisfiable. It follows from Theorem 2.19 that all $S \cup R_1, \ \ldots, \ S \cup R_p$ are strongly unsatisfiable. Hence $S \cup R_1 \vdash \bot, \ \ldots, \ S \cup R_p \vdash \bot, \ X_1 \vdash \bot, \ \ldots, \ X_n \vdash \bot$ strongly represents P.

Now assume that P is false. By Definition 2.13, one of $S \cup R$ or $X_1, \ldots, X_n$ is satisfiable. If it is one of the $X_1, \ldots, X_n$, then we are done. Otherwise, by Theorem 2.19, one of $S \cup R_1, \ \ldots, \ S \cup R_p$ is satisfiable, so that in both cases, the set $S \cup R_1 \vdash \bot, \ \ldots, \ S \cup R_p \vdash \bot, \ X_1 \vdash \bot, \ \ldots, \ X_n \vdash \bot$ strongly represents P. □

At this point we have completed the treatment of $\mathrm{Seq}^{\preceq}_{PCL}$. We included it in this paper, because of its close relation to the theorem proving techniques that we introduce in the next section. In fact, the observation that a sequent calculus can be based on strong representation is a direct consequence of the development of theorem proving techniques for PCL.

Example 2.21. *We want to prove that the context of Example 2.9 is strongly valid. First define:*

$$\begin{array}{lcllcl}
A_0 & = & \forall x \ \#H(x) & B_0 & = & \neg\# \ \forall x \ \#H(x) \\
A_1 & = & \forall x \ H(x) \to \#G(x) & B_1 & = & \neg\# \ \forall x \ H(x) \to \#G(x) \\
A_2 & = & \forall x \ H(x) \to \#M(x) & B_2 & = & \neg\# \ \forall x \ H(x) \to \#M(x) \\
A_3 & = & \langle H(s) \rangle \ G(s) & B_3 & = & \neg\# \ \langle H(s) \rangle \ G(s) \\
A_4 & = & \forall x \ [H(x)] \ G(x) \to M(x) & B_4 & = & \neg\# \ \forall x \ [H(x)] \ G(x) \to M(x) \\
G & = & \neg M(s) & B_5 & = & \neg\# M(s)
\end{array}$$

The sequent expansion consists of the following sequents:

$$\begin{array}{ll}
B_0 \vdash \bot & A_0, A_1, A_2, A_3, B_4 \vdash \bot \\
A_0, B_1 \vdash \bot & A_0, A_1, A_2, A_3, A_4, B_5 \vdash \bot \\
A_0, A_1, B_2 \vdash \bot & A_0, A_1, A_2, A_3, A_4, G \vdash \bot \\
A_0, A_1, A_2, B_3 \vdash \bot &
\end{array}$$

The last two sequents in the second column originate from $M^{\vartheta}(s)$. It created two sequents because it is a theorem.

All sequents are provable. We give a proof of $A_0, A_1, A_2, A_3, B_4 \vdash \bot$. We have $B_4 =$

$$\begin{array}{ll}
\neg\# \ \forall x \ [H(x)] \ G(x) \to M(x) & \equiv \\
\neg \Pi x \ \#(\ [H(x)] \ G(x) \to M(x) \ ) & \equiv \\
\Sigma x \ \neg\#(\ [H(x)] \ G(x) \to M(x) \ ). &
\end{array}$$

We have to apply Σ-introduction, which introduces an eigenvariable for x. This results in the sequent

$$A_0, A_1, A_2, A_3, \ \neg\#(\ [H(y)] \ G(y) \to M(y) \ ) \vdash \bot.$$

We have

$$\begin{array}{ll}
\neg\#(\ [H(y)] \ G(y) \to M(y) \ ) & \equiv \\
\neg(\#H(y) \otimes \neg H(y) \vee \#(\ G(y) \to M(y) \ )) & \equiv \\
\neg\#H(y) \oplus \neg(\neg H(y) \vee \#(\ G(y) \to M(y) \ )). &
\end{array}$$

We apply $\oplus$-introduction. The left sequent $A_0, A_1, A_2, A_3, \neg\#H(y) \vdash \bot$ has an easy proof, which uses only A_0 and $\neg\#H(y)$. The right sequent equals

$$A_0, A_1, A_2, A_3, \neg(\neg H(x) \vee \#(\ G(y) \to M(y)\ )) \vdash \bot.$$

We have

$$\begin{array}{ll} \neg(\neg H(y) \vee \#(\ G(y) \to M(y)\ )) & \equiv \\ H(y) \wedge \neg\#(\ G(y) \to M(y)\ ) & \preceq \\ H(y) \otimes \neg\#(\ G(y) \to M(y)\ ). & \end{array}$$

We apply $\otimes$-introduction, which results in

$$A_0, A_1, A_2, A_3, H(y), \neg\#(\ G(y) \to M(y)\ ) \vdash \bot.$$

The last formula $\preceq \neg\#G(y) \oplus \neg\#M(y)$, so that we end up with the sequents

$$\begin{array}{l} A_0, A_1, A_2, A_3, H(y),\ \neg\#G(y) \vdash \bot \\ A_0, A_1, A_2, A_3, H(y),\ \neg\#M(y) \vdash \bot \end{array}$$

Both sequents are easily provable. The first proof uses A_1, and the second uses A_2.

3 Kleening

In the previous section, we introduced the sequent calculus $\mathrm{Seq}^{\preceq}_{PCL}$. If one wants to establish validity of a context $\|\Gamma\|$, one first has to construct its sequent expansion $\mathrm{Seq}(\|\Gamma\|)$. After that, the resulting sequents can be proven by the rules in Figure 7. Because the deduction rules can be applied only on Kleene operators, the PCL operators have to be replaced by Kleene operators during proof search, which has to be done by the replacement rules in Figure 6. We have shown that these replacements can be applied on the top level of formulas because they preserve strong representation of strong validity of the original context $\|\Gamma\|$.

In the current section, we will show that there is no need to restrict application of the rules of Figure 6 to the top level of formulas. This makes it possible to remove all PCL operators from a formula before reasoning starts. We call this process *Kleening*. After that, we show that $\preceq$ can be further used to tranform Kleene formulas into a form that is almost classical logic. We call this transformation *radicalization*. This transformation is not only an essential part of theorem proving procedures for PCL, it also clarifies the relation between PCL, Kleene logic and classical logic. In particular, radicalization shows that Kleene logic is almost the same as classical logic, which can be interpreted as: Although Kleene logic has more truth values than classical logic, its operators are not changed. As a consequence, one can express definedness/undefinedness in Kleene logic, but there is no mechanism that checks that only defined values are used. If one assumes an ill-typed axiom in Kleene logic, it will be possible to derive consequences from it, which is impossible in PCL.

We turn our attention to Kleening. Kleening is possible, because nearly all operators are $\leq$-monotone.

Theorem 3.1. *All of the logical operators of PCL and the Kleene operators, with the exception of #, are monotone relative to $\preceq$. More precisely:*

- *If $A \preceq B$, then $\neg A \preceq \neg B$.*
- *If $A_1 \preceq B_1$ and $A_2 \preceq B_2$, then*

$$A_1 \wedge A_2 \preceq B_1 \wedge B_2, \quad A_1 \vee A_2 \preceq B_1 \vee B_2, \quad A_1 \to A_2 \preceq B_1 \to B_2,$$
$$A_1 \leftrightarrow A_2 \preceq B_1 \leftrightarrow B_2, \quad [A_1]A_2 \preceq [B_1]B_2, \quad \langle A_1\rangle A_2 \preceq \langle B_1\rangle B_2.$$

- *If $P[x] \preceq Q[x]$, then*

$$\forall x\ P[x] \preceq \forall x\ Q[x], \quad \exists x\ P[x] \preceq \exists x\ Q[x],$$
$$\Pi x\ P[x] \preceq \Pi x\ Q[x], \quad \Sigma x\ P[x] \preceq \Sigma x\ Q[x].$$

Proof. The propositional operators can be checked by truth tables. We give proofs for the two quantifiers $\forall$ and Π. The other two quantifiers can be checked by using the fact that $\exists x\ P[x] \equiv \neg\forall x\ \neg P[x]$ and $\Sigma x\ P[x] \equiv \neg\Pi x\ \neg P[x]$.

Assume that $P[x] \preceq Q[x]$. Let $I = (D, \mathbf{f}, \mathbf{e}, \mathbf{t}, [\ ])$ be an interpretation.

First assume that $I(\ \forall x\ P[x]\ ) = \mathbf{t}$. This means that for every $d \in D$, we have $I^x_d(P[x]) = \mathbf{t}$. Since $P[x] \preceq Q[x]$, we also have $I^x_d(Q[x]) = \mathbf{t}$ for every $d \in D$. This implies that $I(\ \forall x\ Q[x]\ ) = \mathbf{t}$.

Now assume that $I(\ \forall x\ P[x]\ ) = \mathbf{f}$. This means that there is a $d \in D$, s.t. $I^x_d(P[x]) = \mathbf{f}$, and for every $d \in D$, $I^x_d(P[x]) \in \{\mathbf{f}, \mathbf{t}\}$.

Since $P[x] \preceq Q[x]$, the same must hold for $Q[x]$: There is a $d \in D$, s.t. $I^x_d(Q[x]) = \mathbf{f}$, and for every $d \in D$, $I^x_d(Q[x]) \in \{\mathbf{f}, \mathbf{t}\}$. This in turn implies that $I(\ \forall x\ Q[x]\ ) = \mathbf{f}$.

Next assume that $I(\ \Pi x\ P[x]\ ) = \mathbf{t}$. This means that for every $d \in D$, $I^x_d(P[x]) = \mathbf{t}$. Since $P[x] \preceq Q[x]$, it follows that for every $d \in D$, $I^x_d(Q[x]) = \mathbf{t}$, which in turn implies that $I(\ \Pi x\ Q[x]\ ) = \mathbf{t}$.

Finally, assume that $I(\ \Pi x\ P[x]\ ) = \mathbf{f}$. This means that there is a $d \in D$, s.t. $I(P[x]) = \mathbf{f}$. Since $P[x] \preceq Q[x]$, we also have $I(Q[x]) = \mathbf{f}$. This in turn implies that $I(\ \Pi x\ Q[x]\ ) = \mathbf{f}$, so that the proof is complete. □

It can be easily seen that # is not monotone: For example, one has $E \preceq \top$, but not $\#E \preceq \#\top$. Since we have now established that the rules in Figure 6 are not restricted to the top level of a formula, we can use them to fully rewrite a formula into its normal form. This leads to the following definition:

Definition 3.2. *For a formula A in PCL (possibly mixed with Kleene opera-*

tors), we define the Kleening, $\mathrm{Kl}(A)$, *as follows:*

$$
\begin{array}{lcl}
\mathrm{Kl}(\ A\ ) & = & A,\ \textit{if } A \textit{ is a non-equality atom} \\
\mathrm{Kl}(\ \#A\ ) & = & \mathrm{Kl}^{\#}(A) \\
\mathrm{Kl}(\ \bot\ ) & = & \bot \\
\mathrm{Kl}(\ E\ ) & = & E \\
\mathrm{Kl}(\ \top\ ) & = & \top \\
\mathrm{Kl}(\ \neg A\ ) & = & \neg\mathrm{Kl}(A) \\
\mathrm{Kl}(\ A \vee B\ ) & = & \mathrm{Kl}(A) \oplus \mathrm{Kl}(B) \\
\mathrm{Kl}(\ A \wedge B\ ) & = & \mathrm{Kl}(A) \otimes \mathrm{Kl}(B) \\
\mathrm{Kl}(\ A \to B\ ) & = & \neg\mathrm{Kl}(A) \oplus \mathrm{Kl}(B) \\
\mathrm{Kl}(\ A \leftrightarrow B\ ) & = & (\ \neg\mathrm{Kl}(A) \oplus \mathrm{Kl}(B)\ ) \otimes (\ \mathrm{Kl}(A) \oplus \neg\mathrm{Kl}(B)\ ) \\
\mathrm{Kl}(\ \langle A\rangle B\ ) & = & \mathrm{Kl}(A) \otimes \mathrm{Kl}(B) \\
\mathrm{Kl}(\ [A]B\ ) & = & \neg\mathrm{Kl}(A) \oplus \mathrm{Kl}(B) \\
\mathrm{Kl}(\ \forall x\ P[x]\ ) & = & \Pi x\ \mathrm{Kl}(P[x]) \\
\mathrm{Kl}(\ \exists x\ P[x]\ ) & = & \Sigma x\ \mathrm{Kl}(P[x]) \\
\mathrm{Kl}(\ t_1 \approx t_2\ ) & = & t_1 \approx t_2
\end{array}
$$

$$
\begin{array}{lcl}
\mathrm{Kl}^{\#}(\ A\ ) & = & \#A,\ \textit{if } A \textit{ is a non-equality atom} \\
\mathrm{Kl}^{\#}(\ \#A\ ) & = & \top \\
\mathrm{Kl}^{\#}(\ \bot\ ) & = & \top \\
\mathrm{Kl}^{\#}(\ E\ ) & = & \bot \\
\mathrm{Kl}^{\#}(\ \top\ ) & = & \top \\
\mathrm{Kl}^{\#}(\ \neg A\ ) & = & \mathrm{Kl}^{\#}(A) \\
\mathrm{Kl}^{\#}(\ A \vee B\ ) & = & \mathrm{Kl}^{\#}(A) \otimes \mathrm{Kl}^{\#}(B) \\
\mathrm{Kl}^{\#}(\ A \wedge B\ ) & = & \mathrm{Kl}^{\#}(A) \otimes \mathrm{Kl}^{\#}(B) \\
\mathrm{Kl}^{\#}(\ A \to B\ ) & = & \mathrm{Kl}^{\#}(A) \otimes \mathrm{Kl}^{\#}(B) \\
\mathrm{Kl}^{\#}(\ A \leftrightarrow B\ ) & = & \mathrm{Kl}^{\#}(A) \otimes \mathrm{Kl}^{\#}(B) \\
\mathrm{Kl}^{\#}(\ \langle A\rangle B\ ) & = & \mathrm{Kl}^{\#}(A) \otimes (\ \neg\mathrm{Kl}(A) \oplus \mathrm{Kl}^{\#}(B)\ ) \\
\mathrm{Kl}^{\#}(\ [A]B\ ) & = & \mathrm{Kl}^{\#}(A) \otimes (\ \neg\mathrm{Kl}(A) \oplus \mathrm{Kl}^{\#}(B)\ ) \\
\mathrm{Kl}^{\#}(\ \forall x\ P[x]\ ) & = & \Pi x\ \mathrm{Kl}^{\#}(P[x]) \\
\mathrm{Kl}^{\#}(\ \exists x\ P[x]\ ) & = & \Pi x\ \mathrm{Kl}^{\#}(P[x]) \\
\mathrm{Kl}^{\#}(\ \ t_1 \approx t_2\ ) & = & \top
\end{array}
$$

The additional function $\mathrm{Kl}^{\#}$ is used on formulas that are inside the scope of $\#$. Using Theorem 2.18 and Theorem 3.1, the following is easy to prove:

Theorem 3.3. *For every formula* A, $\quad A \preceq \mathrm{Kl}(A)$ *and* $\#A \preceq \mathrm{Kl}^{\#}(A)$.

Definition 3.4. *We say that a formula* A *is* in Kleene logic *if it contains only logical operators from* $\bot, \top, E, \neg, \#, \otimes, \oplus, \Pi, \Sigma$ *(and* $\approx$*), and the operator* $\#$ *is applied only on non-equality atoms.*

It is easily checked that the results of $\mathrm{Kl}(A)$ and $\mathrm{Kl}^{\#}(A)$ are always in Kleene logic. The rules for $\mathrm{Kl}(\ A \leftrightarrow B\ )$, $\mathrm{Kl}^{\#}(\ \langle A\rangle B\ )$, and $\mathrm{Kl}^{\#}(\ [A]B\ )$ may cause an exponential increase in size of the formula. This problem can be avoided by

using suitable subformula replacement rules, which are out of the scope of the current paper.

During the rest of the tranformation, it is convenient to push negation inwards as far as possible. This has the advantage that the polarity of non-trivial subformulas is always positive, which will simplify the remaining transformations.

Definition 3.5. *Let A be a Kleene formula. We say that A is* in negation normal form (NNF) *if negation is applied only on atoms and on formulas of form $\#p(t_1, \ldots, t_n)$.*

Definition 3.6. *We recursively define a function $\mathrm{NNF} = \mathrm{NNF}^+$, which transforms a formula into negation normal form:*

$$
\begin{array}{lcl}
\mathrm{NNF}^+(A) & = & A \text{ if } A \text{ is a non-equality atom} \\
\mathrm{NNF}^+(\#A) & = & \#A \\
\mathrm{NNF}^+(\neg A) & = & \mathrm{NNF}^-(A) \\
\mathrm{NNF}^+(\bot) & = & \bot \\
\mathrm{NNF}^+(E) & = & E \\
\mathrm{NNF}^+(\top) & = & \top \\
\mathrm{NNF}^+(A \otimes B) & = & \mathrm{NNF}^+(A) \otimes \mathrm{NNF}^+(B) \\
\mathrm{NNF}^+(A \oplus B) & = & \mathrm{NNF}^+(A) \oplus \mathrm{NNF}^+(B) \\
\mathrm{NNF}^+(\Pi x\ P[x]) & = & \Pi x\ \mathrm{NNF}^+(P[x]) \\
\mathrm{NNF}^+(\Sigma x\ P[x]) & = & \Sigma x\ \mathrm{NNF}^+(P[x]) \\
\mathrm{NNF}^+(t_1 \approx t_2) & = & t_1 \approx t_2 \\
\\
\mathrm{NNF}^-(A) & = & \neg A \text{ if } A \text{ is a non-equality atom} \\
\mathrm{NNF}^-(\#A) & = & \neg\#A \\
\mathrm{NNF}^-(\neg A) & = & \mathrm{NNF}^+(A) \\
\mathrm{NNF}^-(\bot) & = & \top \\
\mathrm{NNF}^-(E) & = & E \\
\mathrm{NNF}^-(\top) & = & \bot \\
\mathrm{NNF}^-(A \otimes B) & = & \mathrm{NNF}^-(A) \oplus \mathrm{NNF}^-(B) \\
\mathrm{NNF}^-(A \oplus B) & = & \mathrm{NNF}^-(A) \otimes \mathrm{NNF}^-(B) \\
\mathrm{NNF}^-(\Pi x\ P[x]) & = & \Sigma x\ \mathrm{NNF}^-(P[x]) \\
\mathrm{NNF}^-(\Sigma x\ P[x]) & = & \Pi x\ \mathrm{NNF}^-(P[x]) \\
\mathrm{NNF}^-(t_1 \approx t_2) & = & \neg(t_1 \approx t_2)
\end{array}
$$

Before we continue, we give two examples:

Example 3.7. *We Kleene some of the formulas in Example 2.21. We start with* $\mathrm{Kl}(B_0) = \mathrm{Kl}(\neg\#\forall x\ \#H(x)) = \neg\mathrm{Kl}(\#\forall x\ \#H(x)) = \neg\mathrm{Kl}^{\#}(\forall x\ \#H(x)) = \neg\Pi x\ \mathrm{Kl}^{\#}(\#H(x)) = \neg\Pi x\ \top$. *The NNF of this formula equals* $\Sigma x\ \bot$.

We have $\mathrm{Kl}(B_1) = \mathrm{Kl}(\neg\#(\forall x\ H(x) \to \#G(x))) = \neg\mathrm{Kl}^{\#}(\forall x\ H(x) \to \#G(x)) = \neg\Pi x\ \mathrm{Kl}^{\#}(H(x) \to \#G(x)) = \neg\Pi x\ (\mathrm{Kl}^{\#}(H(x)) \otimes \mathrm{Kl}^{\#}(\#G(x))) = \neg\Pi x\ (\#H(x) \otimes \top)$. *The NNF of this formula equals* $\Sigma x\ (\neg\#H(x) \oplus \bot)$.

$\mathrm{Kl}(A_3) = \mathrm{Kl}(\ \langle H(s)\rangle\ G(s)\ ) = H(s) \otimes G(s)$.

$\mathrm{Kl}(A_4) = \mathrm{Kl}(\ \forall x\ [H(x)]\ G(x) \to M(x)) = \Pi x\ (\neg H(x) \oplus \neg G(x) \oplus M(x)\ )$.

$\mathrm{Kl}(B_4) = \mathrm{Kl}(\neg\#(\forall x\ [H(x)]\ G(x) \to M(x))) =$
$\neg\Pi x\ \mathrm{Kl}^{\#}(\ [H(x)]\ G(x) \to M(x)) =$
$\neg\Pi x\ (\mathrm{Kl}^{\#}(H(x)) \otimes (\neg\mathrm{Kl}(H(x)) \oplus \mathrm{Kl}^{\#}(G(x) \to M(x)))\ ) =$
$\neg\Pi x\ (\#H(x) \otimes (\neg H(x) \oplus (\#G(x) \otimes \#M(x)))\ )$. *The NNF of this formula equals* $\Sigma x\ (\neg\#H(x) \oplus (H(x) \otimes (\neg\#G(x) \oplus \neg\#M(x))))$.

Example 3.8. *We apply Kleening and compute the negation normal form of each of the formulas in Example 2.21.*

$$
\begin{array}{lcl@{\qquad}lcl}
A_0 & \preceq & \Pi x\ \#H(x) & B_0 & \preceq & \Sigma x\ \bot \\
A_1 & \preceq & \Pi x\ (\neg H(x) \oplus \#G(x)) & B_1 & \preceq & \Sigma x\ (\neg\#H(x) \oplus \bot) \\
A_2 & \preceq & \Pi x\ (\neg H(x) \oplus \#M(x)) & B_2 & \preceq & \Sigma x\ (\neg\#H(x) \oplus \bot) \\
A_3 & \preceq & H(s) \otimes G(s) & B_3 & \preceq & \neg\#H(s) \oplus (H(s) \otimes \neg\#G(s)) \\
A_4 & \preceq & \Pi x\ (\neg H(s) \oplus & B_4 & \preceq & \Sigma x\ (\neg\#H(x) \oplus (H(x) \otimes \\
 & & \quad \neg G(s) \oplus M(s)) & & & \quad (\neg\#G(x) \oplus \neg\#M(x)))) \\
G & \preceq & \neg M(s) & B_5 & \preceq & \neg\#M(s)
\end{array}
$$

As a consequence, a PCL context can be translated into a set of sequents in Kleene logic, which strongly represents its strong validity. Kleening has a surprising feature, namely that it forgets type information.

Example 3.9. *Consider*

$$
\begin{array}{lcl}
C_1 & = & \forall x\ [H(x)]\ G(x) \to M(x) \\
C_2 & = & \forall x\ [H(x) \wedge G(x)]\ M(x) \\
C_3 & = & \forall x\ H(x) \wedge G(x) \to M(x) \\
C_4 & = & \forall x\ H(x) \to [G(x)]\ M(x) \\
C_5 & = & \forall x\ [H(x)]\ [G(x)]\ M(x).
\end{array}
$$

We have $\mathrm{Kl}(C_1) = \mathrm{Kl}(C_2) = \mathrm{Kl}(C_3) = \mathrm{Kl}(C_4) = \mathrm{Kl}(C_5) =$
$\Pi x\ \neg H(x) \oplus \neg G(x) \oplus M(x)$.

The formulas were different in PCL, but Kleening has widened them into the same formula. The formulas still have different meanings in PCL, because the Kleenings of the $\neg\#C_i$ differ. The formula $\neg\#C_i$ occurs in the sequent that represents the type correctness of C_i.

The observation that Kleening removes type information can be reformulated as: Once a formula has been type checked, its type information can be forgotten. Alternatively, one can say: A typechecked formula can be considered equivalent to its Kleening. We will see later in this section, that Kleene logic is very close to classical logic, and that in most cases, the Kleene translation can be assumed to be in classical logic. In that case, one obtains: Once a PCL formula has been type checked, it can be replaced by its relativization in classical logic. This

applies to all of the examples in Section 1. Note that this does not mean that type checking can be avoided in general, because there also exist formulas that will fail type checking.

Kleene logic is still three-valued. We will now show that widening can be further continued up to a point where one almost obtains classical logic. The following definition prepares for this procedure, in which every atom in a Kleene formula will be replaced by a modified atom that always has a definite truth value (a truth value in $\{\mathbf{f}, \mathbf{t}\}$). This can be done without loosing strong representation, because it is a widening step.

Definition 3.10. *For each predicate symbol p with arity n, we define the following abbreviations:*

$$\begin{cases} p_{\emptyset}(t_1, \ldots, t_n) & := & \bot \\ p_{\mathbf{f}}(t_1, \ldots, t_n) & := & \#p(t_1, \ldots, t_n) \otimes \neg p(t_1, \ldots, t_n) \\ p_{\mathbf{e}}(t_1, \ldots, t_n) & := & \neg\#p(t_1, \ldots, t_n) \\ p_{\mathbf{t}}(t_1, \ldots, t_n) & := & \#p(t_1, \ldots, t_n) \otimes p(t_1, \ldots, t_n) \\ p_{\mathbf{f},\mathbf{e}}(t_1, \ldots, t_n) & := & \neg\#p(t_1, \ldots, t_n) \oplus \neg p(t_1, \ldots, t_n) \\ p_{\mathbf{e},\mathbf{t}}(t_1, \ldots, t_n) & := & \neg\#p(t_1, \ldots, t_n) \oplus p(t_1, \ldots, t_n) \\ p_{\mathbf{f},\mathbf{t}}(t_1, \ldots, t_n) & := & \#p(t_1, \ldots, t_n) \\ p_{\mathbf{f},\mathbf{e},\mathbf{t}}(t_1, \ldots, t_n) & := & \top \end{cases}$$

The following can be easily proven:

Theorem 3.11. *For every atom $p(t_1, \ldots, t_n)$, for every $\lambda \subseteq \{\mathbf{f}, \mathbf{e}, \mathbf{t}\}$, for every interpretation I, we have $I(p_\lambda(t_1, \ldots, t_n)) = \mathbf{t}$ iff $I(p(t_1, \ldots, t_n)) \in \lambda$, and $I(p_\lambda(t_1, \ldots, t_n)) = \mathbf{f}$ otherwise.*

It follows from Theorem 3.11 that one always has $I(p_\lambda(t_1, \ldots, t_n)) \in \{\mathbf{f}, \mathbf{t}\}$.

Definition 3.12. *Let A be a Kleene formula in NNF. We recursively define* the radicalization $\mathrm{Rad}(A)$ *of A as follows:*

$$\begin{array}{lcl} \mathrm{Rad}(\ p(t_1, \ldots, t_n)\ ) & = & p_{\mathbf{t}}(t_1, \ldots, t_n) \\ \mathrm{Rad}(\neg p(t_1, \ldots, t_n)\ ) & = & p_{\mathbf{f}}(t_1, \ldots, t_n) \\ \mathrm{Rad}(\#p(t_1, \ldots, t_n)\ ) & = & p_{\mathbf{f},\mathbf{t}}(t_1, \ldots, t_n) \\ \mathrm{Rad}(\#p(t_1, \ldots, t_n)\ ) & = & p_{\mathbf{e}}(t_1, \ldots, t_n) \\ \\ \mathrm{Rad}(\ t_1 \approx t_2\ ) & = & t_1 \approx t_2 \\ \\ \mathrm{Rad}(\top) & = & \top \\ \mathrm{Rad}(E) & = & \bot \\ \mathrm{Rad}(\bot) & = & \bot \\ \mathrm{Rad}(A \otimes B) & = & \mathrm{Rad}(A) \otimes \mathrm{Rad}(B) \\ \mathrm{Rad}(A \oplus B) & = & \mathrm{Rad}(A) \oplus \mathrm{Rad}(B) \\ \\ \mathrm{Rad}(\Pi x\ P[x]) & = & \Pi x\ \mathrm{Rad}(P[x]) \\ \mathrm{Rad}(\Sigma x\ P[x]) & = & \Sigma x\ \mathrm{Rad}(P[x]) \end{array}$$

Radicalization is called 'radicalization' because in the resulting formula, every non-atomic subformula always has a definite truth value. This has a surprising consequence, namely there is no need to use Kleene operators anymore. In the resulting formula, each Kleene operator can be replaced by its corresponding classical (or PCL) operator without changing the truth value of the formula. This implies that in the last four rules of the definition of Rad, one could have used $\wedge, \vee, \forall, \exists$ instead of $\otimes, \oplus, \Pi, \Sigma$.

Theorem 3.13. *For every Kleene formula A in NNF, we have $A \preceq \mathrm{Rad}(A)$.*

Proof. First check (by case analysis) that

$$\begin{array}{lcl} p(t_1,\ldots,t_n) & \preceq & p_{\mathbf{t}}(t_1,\ldots,t_n), \\ \neg p(t_1,\ldots,t_n) & \preceq & p_{\mathbf{f}}(t_1,\ldots,t_n), \\ \#p(t_1,\ldots,t_n) & \preceq & p_{\mathbf{f},\mathbf{t}}(t_1,\ldots,t_n), \\ \neg\#p(t_1,\ldots,t_n) & \preceq & p_{\mathbf{e}}(t_1,\ldots,t_n), \\ E & \preceq & \bot. \end{array}$$

After that, apply Theorem 3.1. □

We give an example of radicalization:

Example 3.14. *Radicalizing the formulas in Example 3.8 gives:*

$$\begin{array}{lcl} A_0 & = & \Pi x\ H_{\mathbf{f},\mathbf{t}}(x) \\ A_1 & = & \Pi x\ (H_{\mathbf{f}}(x) \oplus G_{\mathbf{f},\mathbf{t}}(x)) \\ A_2 & = & \Pi x\ (H_{\mathbf{f}}(x) \oplus M_{\mathbf{f},\mathbf{t}}(x)) \\ A_3 & = & H_{\mathbf{t}}(s) \otimes G_{\mathbf{t}}(s) \\ A_4 & = & \Pi x\ (H_{\mathbf{f}}(s) \oplus G_{\mathbf{f}}(s) \oplus M_{\mathbf{t}}(s)) \\ G & = & M_{\mathbf{f}}(s) \end{array} \qquad \begin{array}{lcl} B_0 & = & \Sigma x\ \bot \\ B_1 & = & \Sigma x\ (H_{\mathbf{e}}(x) \oplus \bot) \\ B_2 & = & \Sigma x\ (H_{\mathbf{e}}(x) \oplus \bot) \\ B_3 & = & H_{\mathbf{e}}(s) \oplus (H_{\mathbf{t}}(s) \otimes G_{\mathbf{e}}(s)) \\ B_4 & = & \Sigma x\ (H_{\mathbf{e}}(x) \oplus (H_{\mathbf{t}}(x) \otimes \\ & & \quad (G_{\mathbf{e}}(x) \oplus M_{\mathbf{e}}(x)))) \\ B_5 & = & M_{\mathbf{e}}(s) \end{array}$$

The resulting sequents still strongly represent strong validity of the context in Example 2.9.

At this point, we are close enough to classical logic, so that it is possible to define a resolution procedure in the standard way. One can Skolemize the radicalized sequents, factor them into clausal normal form, and apply resolution between atoms $p_\lambda(t_1,\ldots,t_n)$ and $p_\mu(u_1,\ldots,u_n)$, when $\lambda \cap \mu = \emptyset$, and the t_i are unifiable with u_i. Since we are interested in geometric logic, for reasons that were explained in [9], we will develope a theorem proving procedure for geometric logic in the remaining section of this paper. In Section 6 we will describe the remaining transformation from radicalized sequents into geometric logic.

In the remainder of the current section, we will discuss the relation between radicalized formulas and classical logic. We start by observing the following fact:

Theorem 3.15. *For every Kleene formula A that is in NNF, for every interpretation I, we have $I(\mathrm{Rad}(A)) = \mathbf{t}$ implies $I(A) = \mathbf{t}$.*

Proof. The theorem can be easily proven by induction. For the base cases, we have the following implications, which can be proven using Theorem 3.11:

$$\begin{array}{lcl} I(\ p_{\mathbf{t}}(t_1,\ldots,t_n)) = \mathbf{t} & \Rightarrow & I(\ p(t_1,\ldots,t_n)) = \mathbf{t}, \\ I(\ p_{\mathbf{f}}(t_1,\ldots,t_n)) = \mathbf{t} & \Rightarrow & I(\ \neg p(t_1,\ldots,t_n)) = \mathbf{t}, \\ I(\ p_{\mathbf{f},\mathbf{t}}(t_1,\ldots,t_n)) = \mathbf{t} & \Rightarrow & I(\ \#p(t_1,\ldots,t_n)) = \mathbf{t}, \\ I(\ p_{\mathbf{e}}(t_1,\ldots,t_n)) = \mathbf{t} & \Rightarrow & I(\ \neg\#p(t_1,\ldots,t_n)) = \mathbf{t}, \\ I(\bot) = \mathbf{t} & \Rightarrow & I(E) = \mathbf{t}. \end{array}$$

The induction steps for the remaining operators are trivial. □

It follows from Theorem 3.15 and Theorem 3.13 that $I(\mathrm{Rad}(A)) = \mathbf{t} \Leftrightarrow I(A) = \mathbf{t}$, so that, if one uses Kleene logic in the traditional way for satisfiability testing, then it is hardly more expressive than classical logic. The only difference is that Kleene logic has more interpretations, so that it is easier to make formula true. This can only happen if the formula contains occurrences of $\neg\#p(t_1,\ldots,t_n)$.

Lemma 3.16. *Let A be a Kleene formula in NNF. If A does not contain a subformula of form $\neg\#p(t_1,\ldots,t_n)$, then* $\mathrm{Rad}(A)$ *is satisfiable in a two-valued interpretation iff* $\mathrm{Rad}(A)$ *is satisfiable in a three-valued interpretation.*

An example of a set of formulas that is satisfiable in a three-valued interpretation, but not in a two-valued interpretation is the set $\{A_{\mathbf{f},\mathbf{t}},\ A\mathbf{e} \oplus B_{\mathbf{e}}\}$. This set originates from radicalizing the sequent $\{\#A, \neg\#(A \wedge B)\} \vdash \bot$, which results from the incorrect assumption that type correctness of A implies type correctness of $\#(A \wedge B)$.

Atoms of form $\neg\#p(t_1,\ldots,t_n)$ rarely occur in formulas that were introduced by an $E_\vartheta(\|\Gamma\|, i)$ of Definition 2.12. These are the sequents that correspond to the correctness of a theorem. Atoms of this mostly occur in sequents that originate from type checking assumptions and theorems. It follows that most theorems, once they have been type checked, become equivalent to their relativizations into classical logic.

In Example 3.8, the last sequent $A_0, A_1, A_2, A_3, A_4, G \vdash \bot$, which corresponds to correctness of the theorem $M^\vartheta(s)$, does not contain a subformula of form $\neg\#p(t_1,\ldots,t_n)$, while all of the other sequents do, with the exception of $B_0 \vdash \bot$. The reason that B_0 does not contain a formula of this form is the fact that Kl replaced $\neg\#\#H(x)$ by $\bot$.

In our view, the examples and Lemma 3.16 show that Kleene logic does not have much advantage over classical logic, when the application is based on satisfiability checking only. In [4], Kleene logic is used in a more sophisticated way that is related to PCL, and with a similar philosophy: Formulas in which preconditions are violated should never be allowed to denote. This is obtained by introducing two levels: The first level contains only total predicates, and is used to define the preconditions of functions and predicates on the second level. The second level, which contains the theory of interest, is allowed to contain partial functions and predicates, but the preconditions must be expressible by formulas of the first level. Before the second level is considered, all preconditions

must be proven in the theory of the first level. After the preconditions have been checked, by monotonicity of Kleene logic, the second level theory can be treated as a standard, two-valued theory. PCL is more expressive than the approach in [4] in two ways: First, in [4], preconditions in the second level must always have definitions in the first level, which makes that preconditions are always eliminable from the theory. In PCL, it is possible to reason about preconditions involving assumed predicates that have no definitions, like e.g. in the theory $\#A \rightarrow \#(A \vee B)$, which is false in the interpretation $I(A) = \mathbf{t}$, $I(B) = \mathbf{e}$. In [4], $\#A$ and $\#B$ would have to be concrete predicates, which have to be provable. It is not possible to assume that a type condition is true. The second difference is that PCL has an unlimited number of levels of type dependencies, as can be seen from Figure 2, where($(-)$ depends on $\leq$, which in turn depends on N. In [4], at most two levels are possible.

4 Three-Valued Geometric Logic

Geometric logic for theorem proving was introduced in [5]. The proof search algorithm for geometric logic is closely related to model generation (see [15]) or to proof search based on hyper tableaux (see [3]). In [9], we introduced a variant of geometric logic, in which function symbols are replaced by predicates, which is able to deal directly with equality, and which uses learning. The search algorithm is similar to the algorithm in [6]. Whenever the algorithm encounters an existential quantifier, it first tries all existing domain elements as possible witness. If this does not succeed, it extends the model with a new domain element. The difference is that our method replaces function symbols by predicate symbols, and that it relies on lemma learning during search.

We will adapt the strategy of [9] to Kleene logic. As a starting point, consider propositional 2-valued geometric logic. Formulas have form $A_1 \wedge \cdots \wedge A_p \rightarrow B_1 \vee \cdots \vee B_q$, where $A_1, \ldots, A_p, B_1, \ldots, B_q$ are atoms. An interpretation is a finite set of atoms. The search algorithm starts with the empty interpretation $I = \{\}$. During search, it checks for geometric formulas $A_1 \wedge \cdots \wedge A_p \rightarrow B_1 \vee \cdots \vee B_q$, s.t. all $A_i \in I$, but no $B_j \in I$. If no such formula exists, then a model is found. Otherwise, it backtracks through all of the interpretations $I \cup \{B_j\}$. If backtracking exhausts all possibilities, then the algorithm reports that the set of geometric formulas is unsatisfiable.

The search algorithm treats $\mathbf{f}$ and $\mathbf{t}$ differently. Atoms that do not occur in the interpretation, are false by default. In a formula, the atoms on the left hand side are passive, which means that they are waiting until they occur in the interpretation. The atoms on the right hand side are active, which means that they are able to extend the interpretation, when this is necessary. This asymmetry is useful in many practical problems, because problems are often sparse, which means that only a few atoms are true in them.

In Kleene logic, the distinction between passive and active atoms will be more complicated, due to the presence of a third truth value. One first has to decide which truth value will be the default. Both $\mathbf{f}$ and $\mathbf{e}$ are reasonable candidates.

We choose $\mathbf{f}$, because type predicates are naturally false, because it is natural to assume that a given object does not belong to a given type, unless it has to, and also relational translations of functions are naturally false, because functionality restricts them from being true too often. As a consequence, interpretations will list the atoms whose truth assignment is not $\mathbf{f}$. Because there are two possibilities, truth values have to be stored in the interpretation as well.

Inside the formulas, the distinction between passive and active atoms cannot be maintained. An atom of form $A_{\mathbf{e}}$ must be able to extend the interpretation, when it is required, but it must also be able to detect a conflict when $A_{\mathbf{t}}$ is added to the interpretation by another formula. Consider for example the formulas $A_{\mathbf{f}} \vee B_{\mathbf{e}}$ and $A_{\mathbf{f}} \vee B_{\mathbf{t}}$, and the interpretation $I = \{A_{\mathbf{t}}\}$. Looking at the first formula, the atom $A_{\mathbf{f}}$ is inconsistent with I, so that we add $B_{\mathbf{e}}$ to I. Now in the second formula, both $A_{\mathbf{f}}$ and $B_{\mathbf{t}}$ have become inconsistent with I, so that the interpretation is closed, and we have to backtrack. If we would have considered the second formula first, then $B_{\mathbf{t}}$ would have extended the interpretation, and $B_{\mathbf{e}}$ would have been in conflict with it. It follows that atoms of form $B_{\mathbf{e}}$ or $B_{\mathbf{t}}$ can play either a passive or an active role. Atoms of form $B_{\mathbf{f}}, B_{\mathbf{f},\mathbf{e}}, B_{\mathbf{f},\mathbf{t}}$ are always passive, because they are true when no atom of $B_{\mathbf{e}}$ or $B_{\mathbf{t}}$ is present in the interpretation.

Definition 4.1. *A* geometric atom *is an atom of one of the following three forms:*

1. *An atom $p_\lambda(x_1, \ldots, x_n)$, where $x_1, \ldots, x_n$ are variables, not necessarily distinct, and $\lambda = \{\mathbf{f}\}, \{\mathbf{e}\}, \{\mathbf{t}\}, \{\mathbf{f}, \mathbf{e}\}$ or $\{\mathbf{f}, \mathbf{t}\}$.*

2. *An equality atom $x_1 \approx x_2$, with x_1, x_2 distinct variables.*

3. *An existentially quantified atom $\Sigma y\ p_\lambda(x_1, \ldots, x_n)$, where $x_1, \ldots, x_n$ are variables, not necessarily distinct, and $\lambda = \{\mathbf{e}\}$ or $\{\mathbf{t}\}$. There must be at least one occurrence of y among the x_i.*

   We will usually write $\Sigma y\ p_\lambda(x_1, \ldots, x_n, y)$, even when y does not have to be at the last position, and there can be more than one occurrence of y.

A geometric formula *is a formula of form $\Pi \overline{x}\ A_1 \oplus \cdots \oplus A_p$, where each A_i is a geometric atom with all its free variables among $\overline{x}$. It is not required that A_i contains all variables of $\overline{x}$.*

The term 'geometric atom' is slightly misleading because an object of form $\exists y\ p_\lambda(x_1, \ldots, x_n, y)$ is not an atom. We think that it is sufficiently close to an atom, so that it can still be called 'atom'.

The search procedure is an adaptation of the procedure of [9] for classical logic. It is defined in two stages: The first stage is a simple backtracking procedure, that backtracks through all possible interpretations. The second stage is obtained by adding lemma learning to the procedure of the first stage. The addition of lemma learning is essential for obtaining an efficient strategy. In order to define the first stage, we need to define interpretations. We first define the domain elements of the interpretations.

Definition 4.2. *We assume a countably infinite set of domain elements* $\mathcal{E}$.

Definition 4.3. *A* ground substitution Θ *is a partial function from variables to* $\mathcal{E}$. *If* A *is a geometric atom, all whose free variables are defined in* Θ, *then* $A\Theta$ *is the result of replacing each free variable* x *by its corresponding* $x\Theta$.

We call the atoms that can be obtained in this way ground atoms.

As with 'geometric atom', the term 'ground atom' is not completely correct, because 'ground atoms' are not always ground, and also not always atoms, but we think they are close enough.

Definition 4.4. *An* interpretation *is a pair* (E, M) *in which* $E \subseteq \mathcal{E}$ *is a set of elements, and* M *is a set of ground atoms of form* $p_\lambda(e_1, \ldots, e_n)$, *s.t.* $\lambda = \mathbf{e}$ *or* $\lambda = \mathbf{t}$, *and all* $e_i \in E$. *It is not allowed that* M *contains a conflicting pair of form* $p_{\mathbf{e}}(e_1, \ldots, e_n),\ p_{\mathbf{t}}(e_1, \ldots, e_n)$.

An interpretation stores the ground atoms that have truth assignments different from the default value $\mathbf{f}$. Using this semantics, one can define when a ground atom is true in an interpretation. An atom of form $p_{\mathbf{t}}(e)$ is true in M if $p_{\mathbf{t}}(e)$ occurs in M. An atom of form $p_{\mathbf{f}}(e)$ is true in M, if both of $p_{\mathbf{e}}(e)$ and $p_{\mathbf{t}}(e)$ do not occur in M. An atom of form $p_{\mathbf{f},\mathbf{t}}(e)$ is true in M if $p_{\mathbf{e}}(e)$ does not occur in M.

When an atom is not true in an interpretation, there are two possibilities: Either it can be made true by extending M, or it can be made true only by making M smaller. In the second case, we call the atom *in conflict* with M.

Definition 4.5. *Let* (E, M) *be an interpretation. Let* A *be a ground atom with all its elements in* E. *We say that* A *is* false in (E, M) *if one of the following holds:*

1. A *has form* $e_1 \approx e_2$ *and* $e_1 \neq e_2$.
2. A *has form* $p_{\{\lambda\}}(e_1, \ldots, e_n)$, $\lambda \neq \mathbf{f}$, *and* M *does not contain* $p_\lambda(e_1, \ldots, e_n)$.
3. A *has form* $p_\lambda(e_1, \ldots, e_n)$, $\mathbf{f} \in \lambda$, *and* M *contains an atom of form* $p_\mu(e_1, \ldots, e_n)$ *with* $\mu \notin \lambda$.
4. A *has form* $\Sigma y\ p_\lambda(e_1, \ldots, e_n, y)$ *with* $\lambda = \{\mathbf{e}\}, \{\mathbf{t}\}$, *and there is no* $e \in E$, *s.t.* $p_\lambda(e_1, \ldots, e_n, e)$ *occurs in* M.

We say that A *is true in* (E, M) *if* A *is not false in* (E, M). *We say that* A *is* in conflict *with* (E, M) *if one of the following holds:*

1. A *has form* $e_1 \approx e_2$ *and* $e_1 \neq e_2$.
2. A *has form* $p_\lambda(e_1, \ldots, e_n)$, *and* M *contains an atom of form* $p_\mu(e_1, \ldots, e_n)$ *with* $\mu \notin \lambda$.

It can be seen from Definition 4.5 that an atom of form $\Sigma y\ p_\lambda(e_1, \ldots, e_n, y)$ is never in conflict with an interpretation. This is because it is always possible to add a new element e to E, and to add $p_\lambda(e_1, \ldots, e_n, e)$ to M. It is easily shown that a conflict atom is always false:

Lemma 4.6. *Let (E, M) be an interpretation. Let A be a ground atom all whose elements occur in E. If A is in conflict with (E, M), then A is false in (E, M).*

It is also easy to show that the only way of repairing a conflict is by backtracking:

Lemma 4.7. *Let (E, M) be an interpretation. Let A be a ground atom all whose elements occur in E. If A is in conflict with (E, M), then A is also in conflict with every interpretation (E', M'), s.t. $E \subseteq E'$ and $M \subseteq M'$.*

If an atom is false in an interpretation, but not in conflict, then it can be made true by extending the interpretation as follows:

Definition 4.8. *Let (E, M) be an interpretation. Let A be a ground atom with all its elements in E. Assume that A is false in (E, M), but not in conflict with (E, M). We say that A* extends *(E, M) into (E', M') if either*

1. *A has form $p_\lambda(e_1, \ldots, e_n)$ with $\lambda = \{\mathbf{e}\}, \{\mathbf{t}\}$, and $(E', M') = (E, M \cup \{\ p_\lambda(e_1, \ldots, e_n)\ \})$.*
2. *A has form $\Sigma y\ p_\lambda(e_1, \ldots, e_n, y)$ with $\lambda \in \{\mathbf{e}\}, \{\mathbf{t}\}$, and there exists an $e \in E$, s.t. $(E', M') = (\ E,\ M \cup \{\ p_\lambda(e_1, \ldots, e_n, e)\ \})$.*
3. *A has form $\Sigma y\ p_\lambda(e_1, \ldots, e_n, y)$, and there exists an $\hat{e} \notin E$, s.t. $(E', M') = (\ E \cup \{\hat{e}\},\ M \cup \{\ p_\lambda(e_1, \ldots, e_n, \hat{e})\ \})$.*

We write $(E, M) \Rightarrow_A (E', M')$, if A extends (E, M) into (E', M'),

In case A is existential, the relation $\Rightarrow_A$ is non-deterministic, because one can choose either to use an existing $e \in E$ as witness, or to create a new $\hat{e} \notin E$. In the latter case, the actual $\hat{e}$ chosen does not matter. We will always assume that there is a fixed way of obtaining a new $\hat{e} \notin E$, and that only one $\hat{e}$ will be considered.

The following lemmas states that it is always possible to extend, that atoms made true by extension will remain true, and that extension is the smallest modification that makes an atom true.

Lemma 4.9. *Let (E, M) be an interpretation. Let A be a ground atom all whose elements are in E. Assume that A is false in (E, M), but not in conflict with (E, M). Then the following hold:*

1. *There exists an interpretation (E', M'), s.t. $(E, M) \Rightarrow_A (E', M')$.*
2. *A is true in every interpretation (E', M') for which $(E, M) \Rightarrow_A (E', M')$, and in every interpretation (E'', M''), s.t. $E' \subseteq E''$ and $M' \subseteq M''$.*
3. *For every interpretation (E'', M'') with $E \subseteq E''$ and $M \subseteq M''$ in which A is true, there exists an interpretation (E', M'), s.t. $(E, M) \Rightarrow_A (E', M')$ and $E' \subseteq E''$, $M' \subseteq M''$.*

Note that part 3 implies that false atoms that are not in conflict, cannot be made true by backtracking.

Definition 4.10. *Let $F = \Pi\overline{x}\ A_1 \oplus \cdots \oplus A_p$ be a geometric formula. Let Θ be a ground substitution that is defined for all variables in $\overline{x}$. We say that F* is false *in (E, M) with Θ, if all instantiated atoms $A_i\Theta$ are false in (E, M).*

We say that F conflicts *(E, M) with Θ if each of its instantiated atoms $A_i\Theta$ is in conflict with (E, M). In that case, we call F* a conflict formula *of (E, M).*

Using extension, we define the first search algorithm. It tries to extend an interpretation (E, M) into an interpretation (E', M') that makes all formulas true with every ground substitution.

At each stage, the algorithm looks for a formula F and a substitution Θ, s.t. $F\Theta$ is false in the current interpretation. If no F and Θ are found, then (E, M) is a satisfying interpretation. If F is a conflict formula, then the algorithm fails, and it backtracks. If F is not a conflict formula, then the algorithm backtracks through all possible extensions (E', M'), based on an atom A of F that is false, but not in conflict with (E, M).

Definition 4.11. *Algorithm $S_t(\mathcal{G}, E, M)$ is called with a set of geometric formulas $\mathcal{G}$ and an interpretation (E, M). It tries to extend the interpretation into an interpretation (E', M'), that makes all geometric formulas in $\mathcal{G}$ true. If no such interpretation exists, it returns $\bot$.*

If such an extension exists, it either returns an interpretation (E', M') with $E \subseteq E',\ M \subseteq M'$, in which all $\mathcal{G}$ are true, or it does not terminate. $S_t(\mathcal{G}, E, M)$ is defined by case analysis:

MODEL: *If for all formulas $F \in \mathcal{G}$ all ground instances $F\Theta$ that use only elements in E, are true in (E, M), then $S_t(\mathcal{G}, E, M)$ returns (E, M).*

SELECT: *Otherwise, $\mathcal{G}$ contains at least one formula F, for which there exists a substitution Θ, s.t. $F\Theta$ is false in (E, M). Let $A_1, \ldots, A_m$ be the atoms in $F\Theta$ that are not in conflict with (E, M). Select an F and a Θ, for which m is minimal.*

FAIL: *If $m = 0$, then F is a conflict formula, and $S_t(\mathcal{G}, E, M)$ returns $\bot$.*

EXTEND: *If $m > 0$, then for every A_i, for every (E', M') with $(E, M) \Rightarrow_{A_i} (E', M')$, do the following:*

- *Assign $r = S_t(\mathcal{G}, E', M')$. If r is an interpretation, then return r.*

If we reached the end, we know that all recursive calls returned $\bot$. In that case, $S_t(\mathcal{G}, E, M)$ also returns $\bot$.

Theorem 4.12. *Let $\mathcal{G}$ be a set of geometric formulas. Let (E, M) be an interpretation, s.t. for every formula $F \in \mathcal{G}$, every instance $F\Theta$ is true in (E, M). Let $E_0 \subseteq E,\ M_0 \subseteq M$. Then $S_t(\mathcal{G}, E_0, M_0)$ does not return $\bot$.*

Proof. Algorithm S_t is non-deterministic. It is possible that it runs in accordance with (E, M), but it doesn't have to, because it may enter another, possibly infinite branch. We show that it is impossible that it returns $\bot$.

Suppose that $S_t(\mathcal{G}, E_0, M_0)$ returns $\bot$. We will derive a contradiction. Let (E_i, M_i) be the biggest interpretation with $E_i \subseteq E,\ M_i \subseteq M$ that occurred in the run of $S_t(\mathcal{G}, E_0, M_0)$.

Consider $S_t(\mathcal{G}, E_i, M_i)$. Certainly MODEL did not apply, because in that case, $S_t(\mathcal{G}, E_0, M_0)$ would have returned (E_i, M_i). This means that SELECT was entered. Let F, Θ be the formula and the substitution that were selected. Since $F\Theta$ is true in (E, M), there must be a literal A in $F\Theta$, that is true in (E, M). It follows from Lemmas 4.6 and 4.7 that A is not in conflict with (E_i, M_i). As a consequence, $S_t(\mathcal{G}, E_i, M_i)$ will not enter FAIL, and A will be among the A_i. It follows from Lemma 4.9, part 3, that there exists an interpretation (E_{i+1}, M_{i+1}), s.t. $E_i \subseteq E_{i+1} \subseteq E$, and $M_i \subseteq M_{i+1} \subseteq M$. This implies that $S_t(\mathcal{G}, E_{i+1}, M_{i+1})$ was called. Since A is true in (E_{i+1}, M_{i+1}), it follows that $M_i \neq M_{i+1}$, which contradicts the original assumption that (E_i, M_i) is maximal. □

Next we prove completeness. In order to do this, we show that when $S_t(\mathcal{G}, E, M)$ does not return $\bot$, it either constructs or approximates a model. In the proof, we need the assumption that SELECT selects in a fair fashion.

Theorem 4.13. *Let $\mathcal{G}$ be a set of geometric formulas. Let (E, M) be an interpretation. Assume that $S_t(\mathcal{G}, E, M)$ does not return $\bot$. Then there exists an interpretation (E', M') with $E \subseteq E'$ and $M \subseteq M'$, s.t. for all formulas F in $\mathcal{G}$, for all ground substitutions Θ with range in E', the instance $F\Theta$ is true in (E', M').*

Proof. If $S_t(\mathcal{G}, E, M)$ returns through MODEL with interpretation (E', M'), then the theorem is trivially true.
Otherwise, the algorithm runs forever. By König's Lemma, it must pass through an infinite sequence of interpretations $(E_0, M_0), (E_1, M_1), \ldots, (E_i, M_i), \ldots$, s.t. for each i, the next interpretation (E_{i+1}, M_{i+1}) is obtained from (E_i, M_i) by EXTENSION, and (E_0, M_0) is the initial interpretation (E, M). Define $E_\omega = \bigcup E_i$ and $M_\omega = \bigcup M_i$. We show that for every formula F in $\mathcal{G}$, for every ground substitution Θ with range in E_ω, the instance $F\Theta$ is true in (E_ω, M_ω).

Let F, Θ be a formula and a ground substitution with range in E, s.t. $F\Theta$ is false in (E_ω, M_ω). We will derive a contradiction. The atoms in $F\Theta$ can be separated into the atoms $A_1, \ldots, A_m$ that are not in conflict with (E_ω, M_ω), and the atoms $B_1, \ldots, B_n$ that are in conflict with (E_ω, M_ω).

For every B_j, there must exist an interpretation (E_{k_j}, M_{k_j}), s.t. B_j is in conflict with (E_{k_j}, M_{k_j}). Let k be the maximum of the k_j, which must exist because the number of possible B_j is finite. Since $A_1, \ldots, A_m$ are false in (E_ω, M_ω), but not in conflict with it, it follows from Lemma 4.9, part 3, that $A_1, \ldots, A_m$ are false in every interpretation (E_i, M_i). By fairness, there must exist a $k' \geq k$, s.t. F and Θ were selected by $S_t(\mathcal{G}, E_{k'}, M_{k'})$.

If $m = 0$, then $S_t(\mathcal{G}, E_{k'}, M_{k'})$ would have entered FAIL, which contradicts the assumption that $(E_1, M_1), \ldots, (E_i, M_i), \ldots$ is an infinite branch.

Figure 8: Propositional Geometric Formulas

(1) $A_{\mathbf{f},\mathbf{t}}$
(2) $A_{\mathbf{f}} \oplus B_{\mathbf{f},\mathbf{t}}$
(3) $A_{\mathbf{f}} \oplus C_{\mathbf{f},\mathbf{t}}$
(4) $A_{\mathbf{e}} \oplus A_{\mathbf{t}}$
(5) $A_{\mathbf{e}} \oplus B_{\mathbf{e}}$

If $m > 0$, then $(E'_{k'+1}, M_{k'+1})$ must have been obtained from $(E_{k'}, M_{k'})$ by EXTENSION through one of the A_j. This implies, using Lemma 4.9, that A_j is true in every extension of $(E_{k'+1}, M_{k'+1})$, so that it is true in (E_ω, M_ω). This contradicts the assumption that $F\Theta$ is false in (E_ω, M_ω). □

This ends the description of the first algorithm. In the next section, we will add learning to S_t. We end the current section with an example.

Example 4.14. *Assume that we want to refute the sequent* $\#[A](B \wedge C),\ \neg\#[A]B \vdash \bot$. *Kleening the first formula results in* $\#A \otimes (\neg A \oplus (\#B \otimes \#C))$. *Radicalization results in* $A_{\mathbf{f},\mathbf{t}} \otimes (A_{\mathbf{f}} \oplus (B_{\mathbf{f},\mathbf{t}} \otimes C_{\mathbf{f},\mathbf{t}}))$. *This formula can be factored into the first three geometric formulas in Figure 8. Kleening the second formula results in* $\neg\#A \oplus (A \otimes \neg\#B)$. *Radicalization results in* $A_{\mathbf{e}} \oplus (A_{\mathbf{t}} \otimes B_{\mathbf{e}})$. *This formula can be factored into the last two formulas in Figure 8.*

We try to refute the set of formulas using algorithm S_t. *We start with interpretation* (E_0, M_0), *defined by* $E_0 = \{\}$ *and* $M_0 = \{\}$. *Since the example is propositional, we will ignore the ground substitutions. All formulas are true in* (E_0, M_0), *except for the last two formulas* $A_{\mathbf{e}} \oplus A_{\mathbf{t}}$ *and* $A_{\mathbf{e}} \oplus B_{\mathbf{e}}$.

Both formulas are not in conflict with (E_0, M_0). *In the formula* $A_{\mathbf{e}} \oplus A_{\mathbf{t}}$, *both atoms* $A_{\mathbf{e}}$ *and* $A_{\mathbf{t}}$ *are false in* (E_0, M_0) *but not in conflict with* (E_0, M_0). *We have* $(E_0, M_0) \Rightarrow_{A_{\mathbf{e}}} (E_0, M_0 \cup \{A_{\mathbf{e}}\})$, *and* $(E_0, M_0) \Rightarrow_{A_{\mathbf{t}}} (E_0, M_0 \cup \{A_{\mathbf{t}}\})$.

We continue search with $(E_1, M_1) = (\{\}, \{A_{\mathbf{e}}\})$. *Now the first formula* $A_{\mathbf{f},\mathbf{t}}$ *is false in* (E_1, M_1) *and it is in conflict with* (E_1, M_1).

We backtrack and enter the other branch, which results in $(E_2, M_2) = (\{\}, \{A_{\mathbf{t}}\})$. *All formulas are true in* (E_2, M_2), *except for the last formula* $A_{\mathbf{e}} \oplus B_{\mathbf{e}}$.

Atom $A_{\mathbf{e}}$ *is in conflict with* (E_2, M_2). *The other atom* $B_{\mathbf{e}}$ *is false in* (E_2, M_2), *but not in conflict. We have* $(E_2, M_2) \Rightarrow_{B_{\mathbf{e}}} (E_2, M_2 \cup \{B_{\mathbf{e}}\})$.

The resulting interpretation is $(E_3, M_3) = (\{\}, \{A_{\mathbf{t}}, B_{\mathbf{e}}\})$. *Now the second clause* $A_{\mathbf{f}} \oplus B_{\mathbf{f},\mathbf{t}}$ *is false in* (E_3, M_3), *and it is in conflict with* (E_3, M_3).

Since we have exhausted all possibilities, we have shown that the set of formulas is unsatisfiable.

In the next section we give a non-propositional example, in which existential quantification will be used.

5 Lemma Generation

In [9], model search was combined with lemma generation in order to avoid that similar work will be repeated by the search algorithm. In this section, we will show that this is also possible with geometric Kleene logic. Lemma generation for geometric logic works in the same way as for propositional logic. (See [17], [11], [16]). Whenever the search algorithm closes a branch, it constructs a conflict lemma that will prevent that a similar branch will be explored in the future. In the DPLL algorithm, the lemma is constructed by propositional resolution. In geometric logic, the lemma is constructed by rules that are related to predicate resolution. We call the improved algorithm $S_m(\mathcal{G}, E, M)$. It is similar to $S_t(\mathcal{G}, E, M)$, but it always creates a conflict formula when it backtracks. The conflict formulas are derived by the following deduction rules:

Definition 5.1. *We define disjunction resolution and existential resolution:*

Disjunction Resolution: *Let* $F = \Pi\overline{x}\ \Phi(\overline{x}) \oplus p_\lambda(\overline{x})$ *and* $G = \Pi\overline{y}\ \Psi(\overline{y}) \oplus p_{\mu_1}(\overline{y}_1) \oplus \cdots \oplus p_{\mu_n}(\overline{y}_n)$ *be two geometric formulas. Assume that* $n > 0$, *and that for all* j *with* $1 \leq j \leq n$, *we have* $\lambda \cap \mu_j = \emptyset$.

Further assume that the atoms $p(\overline{x})$ *and* $p(\overline{y}_1), \ldots, p(\overline{y}_n)$ *are simultaneously unifiable. Let* Λ *be a most general unifier. Then the formula*

$$\Pi\overline{x}\Lambda\ \Pi\overline{y}\Lambda\ \Phi(\overline{x})\Lambda \oplus \Psi(\overline{y})\Lambda$$

is a disjunction resolvent *of* F *and* G.

Existential Resolution: *Let* $F = \Pi\overline{x}\ \Phi(\overline{x}) \oplus \Sigma y\ p_\lambda(\overline{x}, y)$ *with* $y \notin \overline{x}$ *be a geometric formula. Let* $G =$
$\Pi\overline{z}\ \Pi\overline{v}\ \Psi(\overline{z}) \oplus p_{\mu_1}(\overline{z}_1, \overline{v}_1) \oplus \cdots \oplus p_{\mu_m}(\overline{z}_m, \overline{v}_m) \oplus v_1 \approx z_1 \oplus \cdots \oplus v_n \approx z_n$ *be a geometric formula for which* $\overline{v} \cap \overline{z} = \emptyset$, *and for all* j *with* $1 \leq j \leq m$, *we have* $\lambda \cap \mu_j = \emptyset$.

Assume that there exists a unifier that simultaneously unifies $p(\overline{x}, y)$ *with all* $p(\overline{z}_1, \overline{v}_1), \ldots, p(\overline{z}_m, \overline{v}_m)$, *which merges* y *with all variables in* $\overline{v}$, *but which does not merge* y *with a variable from* $\overline{x}$ *or* $\overline{z}$.

Let Λ *be a most general unifier.*

Then the formula $\Pi\overline{x}\Lambda\ \Pi\overline{z}\Lambda\ \Phi(\overline{x})\Lambda \oplus \Psi(\overline{z})\Lambda \oplus p_\lambda(\overline{x}, z_1)\Lambda \oplus \cdots \oplus p_\lambda(\overline{x}, z_n)\Lambda$ *is an* existential resolvent *of* F *and* G.

Disjunction resolution is closely related to standard resolution. The description of existential resolution is admittedly awkward, but the rule is complicated and we are unfortunately not aware of a simpler way of describing it. In combination with a quantifier, the notation $\overline{x}$ means the set of variables being quantified over. The expression $\Phi(\overline{x})$ denotes a Kleene disjunction with all its variables among $\overline{x}$, but which does not necessarily contain all variables from $\overline{x}$. The notation $p_\lambda(\overline{x}_i)$ denotes a geometric atom, built using variables from $\overline{x}$, but not necessarily containing all variables from $\overline{x}$. The notation x_i denotes a single, arbitrary variable from $\overline{x}$. Variables $x_1, x_2, \ldots$ may be distinct or the same. We give an example of disjunction resolution:

Example 5.2. *Geometric formulas* $\Pi xy\ A_{\mathbf{f}}(x,y) \oplus B_{\mathbf{t}}(x,y)$ *and* $\Pi xyz\ B_{\mathbf{f},\mathbf{e}}(x,y) \oplus B_{\mathbf{e}}(x,z) \oplus C_{\mathbf{t}}(x,y,z)$ *resolve into* $\Pi xy\ A_{\mathbf{t}}(x,y) \oplus C_{\mathbf{t}}(x,y,y)$.

Next we give a sequence of examples of existential resolution, with increasing complexity:

Example 5.3. *Let* F *be the formula* $\Pi xy\ A_{\mathbf{f}}(x,y) \oplus \Sigma z\ B_{\mathbf{t}}(x,y,z)$. *We will show how different formulas* $G_1, \ldots, G_4$ *resolve with* F *using existential resolution.*

In its simplest form, existential resolution is almost the same as disjunction resolution. The formula $G_1 = \Pi xyv\ B_{\mathbf{e}}(x,y,v) \oplus C_{\mathbf{f}}(x,y)$ *can resolve with* F, *which results in* $\Pi xy\ A_{\mathbf{f}}(x,y) \oplus C_{\mathbf{f}}(x,y)$.

G_1 *contains a variable* v, *that is matched with the existentially quantified variable* z *in* F. *It can occur only on positions that match the occurrence of* z *in* F. *Trying to resolve* F *with* $\Pi xy\ B_{\mathbf{e}}(x,y,y) \oplus C_{\mathbf{f}}(x,y)$ *would be incorrect. The formula* $G_2 = \Pi xyv\ B_{\mathbf{e}}(x,y,v) \oplus x \approx v \oplus C_{\mathbf{f}}(x,y)$ *contains one equality involving* v. *In case* $x \approx v$ *holds, the atom* $B_{\mathbf{e}}(x,y,v)$ *need not be true. Hence, we cannot completely resolve* $\Sigma z\ B_{\mathbf{t}}(x,y,z)$ *away. We have to keep the instances of* $B_{\mathbf{t}}(x,y,v)$ *where* $x = v$. *This gives rise to the existential resolvent* $\Pi xy\ A_{\mathbf{f}}(x,y) \oplus C_{\mathbf{f}}(x,y) \oplus B_{\mathbf{t}}(x,y,x)$.

It is possible that the second formula contains more than one variable matching into the existential variable. In formula $G_3 = \Pi xyv_1v_2\ B_{\mathbf{f}}(x,y,v_1) \oplus B_{\mathbf{e}}(x,y,v_2) \oplus x \approx v_1 \oplus y \approx v_2 \oplus C_{\mathbf{f}}(x,y)$, *both variables* v_1 *and* v_2 *match* z. *Existential resolution results in* $\Pi xy\ A_{\mathbf{f}}(x,y) \oplus C_{\mathbf{f}}(x,y) \oplus B_{\mathbf{t}}(x,y,x) \oplus B_{\mathbf{t}}(x,y,y)$.

Finally, there is a special case where no literal matching $B_{\mathbf{t}}(x,y,z)$ *is present in the second formula. This form of existential resolution is called 'degenerated' in [9]. An example is the formula* $G_4 = \Pi xyv\ \ x \approx v \oplus C_{\mathbf{f}}(x,y)$ *which can resolve with* F *either into* $\Pi xyy'\ A_{\mathbf{f}}(x,y) \oplus B_{\mathbf{t}}(x,y,x) \oplus C_{\mathbf{f}}(x,y')$, *or into* $\Pi xyy'\ A_{\mathbf{f}}(x,y) \oplus B_{\mathbf{t}}(x,y,y) \oplus C_{\mathbf{f}}(y,y')$.

The following theorem states that it is always possible, after all alternatives have failed, to construct a conflict formula for the interpretation that we tried to extend.

Theorem 5.4. *Let* F *be a geometric formula. Let* (E,M) *be an interpretation, and let* Θ *be a ground substitution, such that* $F\Theta$ *is false in* (E,M).
Write F *in the form* $\Pi\overline{x}\ \Phi(\overline{x}) \oplus A_1 \oplus \cdots \oplus A_m$, *where* $A_1, \ldots, A_m$ *are the atoms whose instances* $A_i\Theta$ *are not in conflict with* (E,M), *and all atoms in* $\Phi(\overline{x})\Theta$ *are in conflict with* (E,M).

Assume that for every A_j, *for every interpretation* (E',M') *with* $(E,M) \Rightarrow_{A_i\Theta} (E',M')$, *we have a conflict formula. Then it is possible to derive, using disjunction resolution and existential resolution, a conflict formula of* (E,M).

Since Theorem 5.4 is important, but rather complicated, we give an example. The example uses the formulas of Example 5.3.

Example 5.5. *Let the interpretation* (E,M) *be defined by* $E = \{e_0, e_1\}$, $M = \{A_{\mathbf{t}}(e_0,e_1),\ C_{\mathbf{t}}(e_0,e_1)\}$. *Let* $F = \Pi xy\ A_{\mathbf{f}}(x,y) \oplus \Sigma z\ B_{\mathbf{t}}(x,y,z)$ *be as in Example 5.3. Assuming* $\Theta = \{x := e_0,\ y := e_1\}$, *the instance* $F\Theta$ *is false in* (E,M).

We have $(E, M) \Rightarrow_{\Sigma z\ B_{\mathbf{t}}(e_0,e_1,z)} (E_i, M_i)$ *for*

$$\begin{array}{ll} E_0 = \{e_0, e_1\} & M_0 = \{\ A_{\mathbf{t}}(e_0, e_1),\ C_{\mathbf{t}}(e_0, e_1),\ B_{\mathbf{t}}(e_0, e_1, e_0)\ \} \\ E_1 = \{e_0, e_1\} & M_1 = \{\ A_{\mathbf{t}}(e_0, e_1),\ C_{\mathbf{t}}(e_0, e_1),\ B_{\mathbf{t}}(e_0, e_1, e_1)\ \} \\ E_2 = \{e_0, e_1, e_2\} & M_2 = \{\ A_{\mathbf{t}}(e_0, e_1),\ C_{\mathbf{t}}(e_0, e_1),\ B_{\mathbf{t}}(e_0, e_1, e_2)\ \} \end{array}$$

Using recursive calls, algorithm $S_m(\mathcal{G}, E, M)$ *will collect conflict formulas for the interpretations* $(E_0, M_0), (E_1, M_1), (E_2, M_2)$.

Construction of the conflict formula for (E, M) *starts from the conflict formula for* (E_2, M_2). *It has a special form, because of the new element* e_2.

Let G *be the conflict formula that was found for* (E_2, M_2). *There exists a substitution* Θ', *s.t. each atom in* $G\Theta'$ *is in conflict with* (E_2, M_2). *If there is no variable* v *in* G, *for which* $v\Theta' = e_2$, *then each atom in* $G\Theta'$ *is also in conflict with* (E, M), *so that we have a conflict formula for* (E, M). *We can therefore assume that* e_2 *is used by* Θ'. *For simplicity, we assume that* G *contains a single variable* v *with* $v\Theta' = e_2$. *If there are more variables, they can be merged. Since the only occurrence of* e_2 *is in the atom* $B_{\mathbf{t}}(e_0, e_1, e_2)$, *variable* v *can occur only in atoms of form* $B_\lambda(x_i, x_j, v)$ *with* $\mathbf{t} \notin \lambda$, *or of form* $x_i \approx v$. *If an atom* A *containing* v *would have another form, the instance* $A\Theta'$ *would not be in conflict with* (E_2, M_2).

Consider $G_1 = \Pi xyv\ B_{\mathbf{e}}(x, y, v) \oplus C_{\mathbf{f}}(x, y)$ *and* $\Theta' = \{x := e_0,\ y := e_1,\ v := e_2\}$. *The instance* $G_1\Theta'$ *is in conflict with* (E_2, M_2). *The assignment* $v := e_2$ *can be changed to* $v := e_0$ *or* $v := e_1$, *which would make* $B_{\mathbf{f}}(x, y, v)\Theta'$ *in conflict with* $B_{\mathbf{t}}(e_0, e_1, e_0)$ *or* $B_{\mathbf{t}}(e_0, e_1, e_1)$. *Since* G_1 *does contain any other atoms involving* v, *it follows that* G_1 *is also a conflict formula of* (E_0, M_0) *and* (E_1, M_1).

This is reflected by the fact that F *and* G_1 *can resolve into* $\Pi xy\ A_{\mathbf{f}}(x, y) \oplus C_{\mathbf{f}}(x, y)$, *which is a conflict formula of* (E, M) *under the substitution* $\{x := e_0,\ y := e_1\}$.

Next consider $G_2 = \Pi xyv\ B_{\mathbf{e}}(x, y, v) \oplus x \approx v \oplus C_{\mathbf{f}}(x, y)$. *In* G_2, *the variable* v *additionally occurs in the equality* $x \approx v$. *It is still a conflict formula of* (E_2, M_2) *with substitution* Θ'. G_2 *can also be used as conflict formula for* (E_1, M_1) *by changing the assignment of* v *to* $v := e_1$. *It cannot be used as conflict formula for* (E_0, M_0), *because assigning* $v := e_0$ *makes the equality true.*

The existential resolvent of F *and* G_2, $\Pi xy\ A_{\mathbf{f}}(x, y) \oplus C_{\mathbf{f}}(x, y) \oplus B_{\mathbf{t}}(x, y, x)$ *reflects this fact. In order to obtain a conflict clause for* (E, M), *the atom* $B_{\mathbf{t}}(x, y, x)$ *needs to be resolved away. This is possible if we have a conflict formula for* (E_0, M_0). *Assume that* $H = \Pi xy\ x \approx y \oplus B_{\mathbf{f}}(x, y, x)$ *is such a conflict formula with substitution* $\{x := e_0,\ y := e_1\}$. *Using disjunction resolution, one obtains* $\Pi xy\ A_{\mathbf{f}}(x, y) \oplus C_{\mathbf{f}}(x, y) \oplus x \approx y$, *which is a conflict clause for* (E, M).

Example 5.5 shows the general pattern how existential resolution is used. When $S_m(\mathcal{G}, E, M)$ encounters an existential quantifier, it has to find a witness. It first tries the existing elements $e_0, \ldots, e_{n-1}$. If they all fail to produce a satisfying interpretation, it tries a new element e_n. If this also fails, it has conflict formulas for all $e_0, \ldots, e_n$. The conflict formula for e_n is resolved with the formula that contains the existential quantifier. The form of the resolvent depends on the inequalities that occur the conflict formula for e_n. For every inequality, it

contains an additional atom that prevents it from being a conflict formula. The additional atoms are resolved away with disjunction resolution. We now give the proof:

Proof. Theorem 5.4 is proven by induction. Some care has to be given to finding a suitable measure. Existential resolution involving an atom of form $\Sigma y\ B_\lambda(\overline{x}, y)$ may possibly create a new formula containing different atoms of form $B_\lambda(\overline{x}, x_i)$, which have to be resolved away by disjunction resolution.

Let $\mathrm{CLOS}(F, \Theta)$ abbreviate the property: It is possible to derive from F and Θ a formula F' and substitution Θ', s.t. every atom in $F'\Theta'$ is in conflict with (E, M). We prove $\mathrm{CLOS}(F, \Theta)$ by induction as follows: If $F\Theta$ is not by itself in conflict with (E, M), we derive a new F' using disjunction resolution or existential resolution from F in combination with a conflict formula for one of the extensions of (E, M). The new formula F' either contains strictly less existential atoms than F, or it contains the same number of existential atoms, but the number of atoms A for which $A\Theta$ is false in (E, M) but not in conflict, has decreased. Write F in the form $\Pi\overline{x}\ \Phi \oplus A_1 \oplus \cdots \oplus A_m$ as in the statement of the theorem. We will resolve A_m away.

- If A_m is an existential atom, then it has form $\Sigma y\ p_\lambda(\overline{x}, y)$ with $\lambda \in \{\mathbf{e}, \mathbf{t}\}$.

  Choose an $\hat{e} \notin E$. Define $(\hat{E}, \hat{M}) = (E \cup \{\hat{e}\},\ M \cup \{p_\lambda(\overline{x}\Theta, \hat{e})\}\ )$.

  Since $(E, M) \Rightarrow_{A_m\Theta} (\hat{E}, \hat{M})$, and we have assumed existence of a conflict formula for every extension of (E, M), there must exist a formula $\hat{C}$ and a ground substitution $\hat{\Theta}$, s.t. every atom in $\hat{C}\hat{\Theta}$ is in conflict with $(\hat{E}, \hat{M})$. We can assume that $\hat{C}$ and A_m have no variables in common.

  The variables of $\hat{C}$ can be partitioned into $\overline{v}$ and $\overline{z}$, where $v \in \overline{v}$ iff $v\hat{\Theta} = \hat{e}$, and $v \in \overline{z}$ otherwise. It is clear from the construction that $\overline{v}$ and $\overline{z}$ are disjoint. The atoms in $\hat{C}$ can be partitioned into those that contain a variable from $\overline{v}$, and those that do not. Using this, $\hat{C}$ can be written in the form $\Pi\overline{z}\ \overline{v}\ \Psi(\overline{z}) \oplus \Psi'(\overline{z}, \overline{v})$. For every atom $A \in \Psi(\overline{z})$, it must be the case that $A\hat{\Theta}$ is a conflict atom of (E, M).

  If $\overline{v}$ is empty, then $\hat{C}$ is already a conflict formula of (E, M), so that we have established $\mathrm{CLOS}(F, \Theta)$ with $(\hat{C}, \hat{\Theta})$.

  Since every atom A in $\Psi'(\overline{z}, \overline{v})$ contains variables from $\overline{v}$, every $A\hat{\Theta}$ must contain $\hat{e}$. This implies that there are only two types of atom in $\Psi'(\overline{z}, \overline{v})$, namely atoms of form $v \approx z$ with $v \in \overline{v},\ z \in \overline{z}$, and atoms of form $P_\mu(\overline{z}', \overline{v}')$, with $\overline{z}' \subseteq \overline{z},\ \ \overline{v}' \subseteq \overline{v}$, and $\lambda \notin \mu$.

  This implies that $\hat{C}$ can be written in the form

  $$\Pi\overline{z}\ \overline{v}\ \Psi(\overline{z}) \oplus p_{\mu_1}(\overline{z}_1, \overline{v}_1) \oplus \cdots \oplus p_{\mu_m}(\overline{z}_m, \overline{v}_m) \oplus v_1 \approx z_1 \oplus \cdots \oplus v_n \approx z_n,$$

  where $\overline{z}\hat{\Theta}$ does not contain $\hat{e}$, and $\overline{v}\Theta = \{\hat{e}\}$.

  Because the substitution $\Theta \cup \{y := \hat{e}\} \cup \hat{\Theta}$ merges the $p(\overline{z}_i, \overline{v}_i)$ with $p(\overline{x}, y)$, it is clear that $p(\overline{x}, y)$ and the $p(\overline{z}_i, \overline{v}_i)$ have a simultaneous most general

unifier Λ. From the definition of mgu, it follows that there exists a substitution Θ_R, s.t. $\Lambda \cdot \Theta_R = \Theta \cup \{y := \hat{e}\} \cup \hat{\Theta}$.

Since $\overline{v}\hat{\Theta} = \{\hat{e}\}$ and $\hat{e} \notin \overline{z}\hat{\Theta}$, it follows that Λ does not unify a variable in $\overline{v}$ with a variable in $\overline{z}$. Since $y\Lambda = \hat{e}$ as well, it also does not unify y with a variable in $\overline{z}$. It is also not possible that Λ unifies y with a variable in $\overline{x}$, because Θ matches $\overline{x}\Theta \subseteq E$.

At this point, we know that it is possible to construct the existential resolvent $R =$

$$\Pi\overline{x}\Lambda\ \Pi\overline{z}\Lambda\ \Phi(\overline{x})\Lambda \oplus A_1\Lambda \oplus \cdots \oplus A_{m-1}\Lambda \oplus \Psi(\overline{z})\Lambda \oplus p_\lambda(\overline{x}, z_1)\Lambda \oplus \cdots \oplus p_\lambda(\overline{x}, z_n)\Lambda.$$

We will use $\mathrm{CLOS}(R, \Theta_R)$ as induction hypothesis. It is clear that R contains one existential literal less than C, because $\hat{C}$ does not contain existential literals.

We have to show that for every atom A of R, either $A\Theta_R$ is in conflict with (E, M), or for every interpretation (E', M'), with $(E, M) \Rightarrow_{A\Theta_R} (E', M')$ there exists a conflict formula.

If A is in $\Phi(\overline{x})\Lambda$, then there exists a B in $\Phi(\overline{x})$, s.t. $A = B\Lambda$. Since $A\Theta_R = B\Lambda\Theta_R = B\Theta$, and $B\Theta$ is in conflict with (E, M), it follows that $A\Theta_R$ is in conflict with (E, M).

If A is in $\Psi(\overline{z})\Lambda$, then there exists a B in $\Psi(\overline{z})$, s.t. $A = B\Lambda$. We know that $A\Theta_R = B\Lambda\Theta_R = B\hat{\Theta}$. We know by construction of $\Psi(\overline{z})$ that $B\hat{\Theta}$ is in conflict with (E, M). It follows that $A\Theta_R$ is in conflict with (E, M).

If A is one of the $A_i\Lambda$ with $i \leq m-1$, then $A_i\Lambda\Theta_R = A_i\Theta$. If $(E, M) \Rightarrow_{A_i\Lambda\Theta_R} (E', M')$, then clearly $(E, M) \Rightarrow_{A_i\Theta} (E', M')$. We assumed in the statement of the theorem that we have conflict formula for every such interpretation (E', M').

Finally, if A is one of the $p_\lambda(\overline{x}, z_i)\Lambda$, then $A\Theta_R$ has form $p_\lambda(\overline{x}, z_i)\Lambda\Theta_R$. Every argument position that contains the variable z_i must have contained y in the original literal $p_\lambda(\overline{x}, y)$ of F. We know by construction of $\overline{z}$, that $z_i\Lambda\Theta_R = z_i\hat{\Theta} \in E$. Hence, there is an $e \in E$, for which we can write $A\Theta_R = p_\lambda(\overline{x}, z_i)\Lambda\Theta_R$ as $p_\lambda(\overline{x}, y)\Theta\{y := e\}$. If $(E, M) \Rightarrow_{A\Theta_R} (E', M')$, then we have $(E, M) \Rightarrow_{p_\lambda(\overline{x},y)\Theta\{y:=e\}} (E', M')$, which in turn implies $(E, M) \Rightarrow_{(\Sigma y\ p_\lambda(\overline{x},y))\Theta} (E', M')$, so that it it follows from the assumptions of the theorem that there exists a conflict formula.

- If $A_m(\overline{x})$ is a variable atom, then it has form $p_\lambda(\overline{x})$ with $\lambda \in \{\{\mathbf{f}\}, \{\mathbf{e}\}, \{\mathbf{t}\}, \{\mathbf{f}, \mathbf{e}\}, \{\mathbf{f}, \mathbf{t}\}\}$.

  It can be easily checked that $\mathbf{f} \in \lambda$, implies that $p_\lambda(\overline{x}\Theta)$ is either true in (E, M), or in conflict with (E, M). This follows from the fact that (E, M) contains, by Definition 4.4, either exactly one of $p_{\mathbf{e}}(\overline{x}\Theta)$, $p_{\mathbf{t}}(\overline{x}\Theta)$, or neither of them. This contradicts the assumption that A_m is false in (E, M) but not in conflict. Hence $\lambda = \{\mathbf{e}\}$ or $\{\mathbf{t}\}$.

There is a single interpretation (E', M') for which $(E, M) \Rightarrow_{A_m\Theta} (E', M')$, namely $(E', M') = (E, M \cup \{p_\lambda(\overline{x}\Theta)\})$.

From the assumptions of the theorem, there exists a conflict formula $\hat{C}$ for (E', M'). Let $\hat{\Theta}$ be the substitution which brings every atom $A \in \hat{C}\hat{\Theta}$ in conflict with (E', M'). We can assume without loss of generality that $\hat{C}$ has no variables in common with F.

If there is no atom $A \in \hat{C}$, for which $A\hat{\Theta}$ has form $p_\mu(\overline{x}\hat{\Theta})$ with $\lambda \cap \mu = \emptyset$, then $\hat{C}$ is a conflict formula of (E, M), and we have shown $\mathrm{CLOS}(F, \Theta)$ with $(\hat{C}, \hat{\Theta})$.

Otherwise, $\hat{C}$ can be written in the form $\hat{C} = \Pi\overline{y}\ \Psi(\overline{y}) \oplus p_{\mu_1}(\overline{y}_1) \oplus \cdots \oplus p_{\mu_n}(\overline{y}_n)$, where the $p_{\mu_j}(\overline{y}_j)$ are the literals for which $p(\overline{x}\Theta) = p(\overline{y}_j\hat{\Theta})$, $\mu_i \cap \lambda = \emptyset$, and the literals in $\Psi(\overline{y})$ are the remaining literals. It is clear that $p(\overline{x})$ and $p(\overline{y}_1), \ldots, p(\overline{y}_n)$ can be simultaneously unified, so it is possible to construct the disjunction resolvent

$$R = \Pi\overline{x}\Lambda\ \Pi\overline{y}\Lambda\ \Phi(\overline{x})\Lambda \oplus A_1\Lambda \oplus \cdots \oplus A_{m-1}\Lambda \oplus \Psi(\overline{y})\Lambda.$$

By definition of mgu, there exists a ground substitution Θ_R, s.t. $\Lambda \cdot \Theta_R = \Theta \cup \hat{\Theta}$.

We want to use $\mathrm{CLOS}(R, \Theta_R)$ as induction hypothesis. First observe that every atom in $\Phi(\overline{x})\Lambda\Theta_R \cup \Psi(\overline{y})\Lambda\Theta_R = \Phi(\overline{x})\Theta \cup \Psi(\overline{y})\hat{\Theta}$ is in conflict with (E, M).

Secondly, for every atom A_i in $A_1, \ldots, A_{m-1}$, we have $A_i\Lambda\Theta_R = A_i\Theta$. This implies that, using the assumptions of the theorem, we have a conflict formula for every interpretation (E', M') s.t. $(E, M) \Rightarrow_{A_i\Lambda\Theta_R} (E', M')$.

Finally, observe that $A_1\Lambda, \ldots, A_{m-1}\Lambda$ are the atoms A of R, for which A is false in (E, M) but not in conflict. Since $m - 1 < m$, we can use $\mathrm{CLOS}(R, \Theta_R)$ as induction hypothesis.

□

We end the section with an example of a complete refutation by algorithm S_m.

Example 5.6. *Suppose that we want to use geometric Kleene logic to prove that $a \approx b$ implies $s(a) \approx s(b)$. This means that we have to prove the sequent $a \approx b,\ s(a) \not\approx s(b) \vdash \bot$. Using the transformations in Section 6, we obtain the following geometric formulas:*

(1) $\Sigma y\ A_{\mathbf{t}}(y)$
(2) $\Sigma y\ B_{\mathbf{t}}(y)$
(3) $\Pi x\ \Sigma y\ S_{\mathbf{t}}(x, y)$
(4) $\Pi\alpha\beta\ A_{\mathbf{f},\mathbf{e}}(\alpha) \oplus B_{\mathbf{f},\mathbf{e}}(\beta) \oplus \alpha \approx \beta$
(5) $\Pi\alpha\beta\gamma\ A_{\mathbf{f},\mathbf{e}}(\alpha) \oplus B_{\mathbf{f},\mathbf{e}}(\beta) \oplus S_{\mathbf{f},\mathbf{e}}(\alpha, \gamma) \oplus S_{\mathbf{f},\mathbf{e}}(\beta, \gamma)$

The predicate $A(y)$ can be interpreted as $y \approx a$, the predicate $B(y)$ can be interpreted as $y \approx b$, and $S(x, y)$ can be interpreted as as $s(x) \approx y$. The general method of obtaining such translations will be given in Definition 6.2.

We use algorithm S_m. It starts with $(E_1, M_1) = (\{\}, \{\})$. All formulas are true in (E_1, M_1) with every substitution, except for (1) and (2) which are false with the empty substitution. Neither formula is a conflict formula. Assume that S_m selects (1). Then it checks the interpretations (E_2, M_2) for which $(E_1, M_1) \Rightarrow_{\Sigma y\ A_{\mathbf{t}}(y)} (E_2, M_2)$. It is clear that the only candidate is $(E_2, M_2) = (\{e_0\}, \{A_{\mathbf{t}}(e_0)\})$. Proof search continues with (E_2, M_2). Formula (2) is false under the empty substitution. Formula (3) is false with substitution $\{x := e_0\}$. Neither formula is a conflict formula of (E_1, M_1). Assume that S_m selects (2) and the empty substitution. Then we have $(E_2, M_2) \Rightarrow_{\Sigma y\ B_{\mathbf{t}}(y)} (E_3, M_3), (E_4, M_4)$ with

$$\begin{array}{ll} E_3 = \{e_0\} & M_3 = \{A_{\mathbf{t}}(e_0),\ B_{\mathbf{t}}(e_0)\} \\ E_4 = \{e_0, e_1\} & M_4 = \{A_{\mathbf{t}}(e_0),\ B_{\mathbf{t}}(e_1)\} \end{array}$$

Assume that S_m explores (E_3, M_3) first. Formula (3) is false in (E_3, M_3) with substitution $\Theta = \{x := e_0\}$, but not in conflict. All other formulas are true in (E_3, M_3). We have $(E_3, M_3) \Rightarrow_{\Sigma y\ S_{\mathbf{t}}(e_0, y)} (E_4, M_4), (E_5, M_5)$ with

$$\begin{array}{ll} E_4 = \{e_0\} & M_4 = \{A_{\mathbf{t}}(e_0),\ B_{\mathbf{t}}(e_0),\ S_{\mathbf{t}}(e_0, e_0)\} \\ E_5 = \{e_0, e_1\} & M_5 = \{A_{\mathbf{t}}(e_0),\ B_{\mathbf{t}}(e_0),\ S_{\mathbf{t}}(e_0, e_1)\} \end{array}$$

Assume that (E_4, M_4) is explored first. Formula (5) is a conflict formula with substitution $\{\alpha := e_0,\ \beta := e_0,\ \gamma := e_0\}$.

Algorithm S_m backtracks and explores (E_5, M_5). Formula (5) is again a conflict formula, but now with substitution $\{\alpha := e_0,\ \beta := e_0,\ \gamma := e_1\}$. It follows that algorithm S_m will apply existential resolution between (3) and (5), merging $S(x, y)$ with $S(\alpha, \gamma)$ and $S(\beta, \gamma)$. The result is $\Pi x\ A_{\mathbf{f},\mathbf{e}}(x) \oplus B_{\mathbf{f},\mathbf{e}}(x)$, which we call formula (6). It is a conflict formula of (E_3, M_3), which is consistent with Theorem 5.4.

Now S_m has a conflict formula for (E_3, M_3), after which it will consider (E_4, M_4). Formula (4) is a conflict formula with substitution $\{\alpha := e_0,\ \beta := e_1\}$. Since extensions (E_3, M_3) and (E_4, M_4) were based on formula (2), Algorithm S_m will apply disjunction resolution between (2) and (4), matching $B(y)$ with $B(\beta)$. The result is $\Pi x\ A_{\mathbf{f},\mathbf{e}}(x) \oplus B_{\mathbf{t}}(x)$. Since formula (4) contains an equality, the resolvent is not yet a conflict formula for (E_2, M_2). In order to obtain a conflict formula, S_m has to apply further disjunction resolution with formula (6). The result is $\Pi x\ A_{\mathbf{f},\mathbf{e}}(x)$, which we will call formula (7). It is a conflict formula for (E_2, M_2) with substitution $\{x := e_0\}$. One final step of existential resolution with formula (1) yields $\bot$, which is a conflict formula of $(\{\}, \{\})$.

6 Transformation to Geometric Logic

In the previous two sections we have developed a sound and complete strategy for checking satisfiability of sets of geometric Kleene formulas. It remains to provide a transformation that replaces sequents by sets of geometric formulas. When applied to a set of sequents, the transformation has to preserve strong representation. A large part of the transformation was already given in Section 3.

Applying the sequence Kl; NNF; Rad removes PCL operators, transforms into NNF, and radicalizes the atoms in the formulas, after which all logical operators in the formula can be considered classical. As a consequence, the resulting formula sets are so close to classical logic that the transformations of [9] can be used with small adaptations.

If one applies transformation Kl naively, it may cause an exponential increase in formula size. In order to avoid this, it is possible to define subformula replacement. Subformula replacement is a standard technique used in first-order logic theorem proving, in which a complex subformula of form $A(\overline{x})$ is replaced by a new name $p(\overline{x})$, together with a definition $\forall \overline{x}\ p(\overline{x}) \leftrightarrow A(\overline{x})$, in which $\overline{x}$ are the free variables of $A(\overline{x})$. For a discussion of subformula replacement techniques for first-order logic, we refer to [18]. Subformula replacement in PCL is more complicated than in first-order logic, because it has to take place before radicalization, at a point where PCL still significantly differs from classical logic. We will not discuss this further in the present paper, because of length restrictions. In the current section, a restricted form of subformula replacement will be used, which is completely standard, because it takes place after radicalization.

We now describe the rest of the transformation. The main step is the replacement of function symbols by relation symbols. The reason for doing this is that we want a procedure that is able to combine proof search with model search, which is done more naturally in a function-free setting. In order to eliminate a function symbol f with arity n, we introduce a relation symbol P_f with arity $n+1$. Constants are treated as function symbols with arity 0.

Example 6.1. *Consider the sequent*

$$\left.\begin{array}{l} \forall x\ \#N(x) \\ N(0) \\ \forall x\ N(x) \to N(s(x)) \\ \neg N(s(s(0))) \end{array}\right\} \vdash \bot$$

Applying Kl; NNF; Rad *results in*

$$\left.\begin{array}{l} \Pi x\ N_{\mathbf{f},\mathbf{t}}(x) \\ N_{\mathbf{t}}(0) \\ \Pi x\ N_{\mathbf{f}}(x) \oplus N_{\mathbf{t}}(s(x)) \\ N_{\mathbf{f}}(s(s(0))) \end{array}\right\} \vdash \bot$$

The sequent contains two functions symbols 0 *and* s*. We introduce a predicate symbol* Z *with arity* 1 *and a predicate symbol* S *with arity* 2.

$$\left.\begin{array}{l} \Sigma y\ Z_{\mathbf{t}}(y) \\ \Pi x\ \Sigma y\ S_{\mathbf{t}}(x,y) \\ \\ \Pi x\ N_{\mathbf{f},\mathbf{t}}(x) \\ \Pi z\ Z_{\mathbf{f},\mathbf{e}}(z) \oplus N_{\mathbf{t}}(z) \\ \Pi x z\ S_{\mathbf{f},\mathbf{e}}(x,z) \oplus N_{\mathbf{f}}(x) \oplus N_{\mathbf{t}}(z) \\ \Pi z_1 z_2 z_3\ Z_{\mathbf{f},\mathbf{e}}(z_1) \oplus S_{\mathbf{f},\mathbf{e}}(z_1,z_2) \oplus S_{\mathbf{f},\mathbf{e}}(z_2,z_3) \oplus N_{\mathbf{f}}(z_3) \end{array}\right\} \vdash \bot$$

The resulting formulas are in geometric form, but this is not always the case.

In the last sequent, the first two formulas ensure that the introduced predicates are total. There is no need to add functionality axioms because the predicates occur only negatively in the remaining formulas. This implies that in every interpretation, the predicates can always be made functional without making any formulas false.

We now introduce the transformation formally, and prove its correctness. We call it *anti-Skolemization*, because it does the opposite of Skolemization. Anti-Skolemization for Kleene logic does not differ from the procedure in [9] or [2].

Definition 6.2. *We assume a mapping that maps every n-arity function symbol f to a unique $(n+1)$-arity predicate symbol $\overline{F}$.*

Let A be a Kleene formula that is in NNF and radicalized. An anti-Skolemization *of A is a formula that is the result of making the following replacement as long as A contains functional terms:*

Select a functional term $f(x_1, \ldots, x_n)$ in which all $x_1, \ldots, x_n$ are variables. Such a term necessarily exists. It is possible that $n = 0$.

Write A in the form $A[\ B[\ f(x_1, \ldots, x_n)\ ]\ ]$, where B is a subformula of A that contains at least one of the occurrences of $f(x_1, \ldots, x_n)$. Replace $A[\ B[\ f(x_1, \ldots, x_n)\ ]\ ]$ by $A[\ \Pi z\ \overline{F}_{\mathbf{f},\mathbf{e}}(x_1, \ldots, x_n, z) \oplus B[z]\ ]$.

Definition 6.2 leaves some choice for heuristics in the choice of B, as is shown by the next example:

Example 6.3. *Consider the formula $A = \Pi x\ P_{\mathbf{t}}(s(x)) \otimes Q \otimes R_{\mathbf{t}}(s(x))$. If one chooses $B = P_{\mathbf{t}}(s(x)) \otimes Q \otimes R_{\mathbf{t}}(s(x))$, then one immediately obtains $\Pi x z\ S_{\mathbf{f},\mathbf{e}}(x, z) \oplus (P_{\mathbf{t}}(z) \otimes Q \otimes R_{\mathbf{t}}(z))$. If one first chooses $B = P_{\mathbf{t}}(s(x))$, and after that $B' = R_{\mathbf{t}}(s(x))$, one obtains $\Pi x\ (\Pi z\ S_{\mathbf{f},\mathbf{e}}(x, z) \oplus P_{\mathbf{t}}(z)) \otimes Q \otimes (\Pi z\ S_{\mathbf{f},\mathbf{e}}(x, z) \oplus R_{\mathbf{t}}(z))$.*

The first method has the advantage that $S_{\mathbf{f},\mathbf{e}}(x, z)$ is introduced only once. It has the disadvantage that it causes Q to depend on $S_{\mathbf{f},\mathbf{e}}(x, z)$. The second choice does not have this problem, but it introduces $S_{\mathbf{f},\mathbf{e}}(x, z)$ twice. In the presence of big terms, this may lead to long, repeated sequences of atoms.

A compromise could be to take B as the largest purely disjunctive subformula that contains $f(x_1, \ldots, x_n)$, or to factor A into clauses first, and to apply anti-Skolemization on clauses only. It may also be useful to simplify formulas before anti-Skolemization.

We postpone the correctness proof, because we first want to give the complete transformation. At this point, the formulas are almost geometric. It remains to do some factoring, to do some simplifications, to substitute negative equalities away, and to rename existentially quantified subformulas that are not an atom. We define the last step of the transformation on sequents, because it can split a single formula into multiple formulas.

Definition 6.4. *Let $S \vdash \bot$ be a sequent containing only Kleene formulas that are in NNF, function free, and radicalized. We define the final transformation* Geo *that makes the formulas in S geometric:*

1. *Normalize every formula A in S using the rewrite rules in Figure 9.*

2. *Replace every formula in S of form $A \otimes B$ by A and B separately.*

3. *As long as S contains a formula A that contains an existentially quantified subformula $\Sigma y\ P[y]$ for which $P[y]$ is not a geometric atom or does not contain y, write A in the form $A[\ \Sigma y\ P[x_1, \ldots, x_n, y]\ ]$, where $x_1, \ldots, x_n$ are the other free variables of P. After that, replace $A[\ \Sigma y\ P[x_1, \ldots, x_n, y]\ ]$ by $A[\ \Sigma y\ p_{\mathbf{t}}(x_1, \ldots, x_n, y)\ ]$ and $\Pi x_1 \cdots x_n\ \Pi y\ p_{\mathbf{f},\mathbf{e}}(x_1, \ldots, x_n, y) \oplus P[x_1, \ldots, x_n, y]$, using a new predicate symbol p.*

At this point, every formula in S has form

$$\Pi \overline{x}\ \bigoplus \overline{A},$$

where $\overline{x}$ is a set of variables, and each element of $\overline{A}$ has one of the following five forms:

1. *A geometric atom. (See Definition 4.1.)*

2. *A negative equality $x_1 \not\approx x_2$ with $x_1, x_2 \in \overline{x}$.*

3. *A positive equality of form $x \approx x$ with $x \in \overline{x}$. (Positive equalities with distinct variables are geometric atoms.)*

4. *$\bot$.*

5. *$\top$.*

For every formula $\Pi \overline{x}\ \bigoplus \overline{A}$ in S, we proceed as follows: If $x \approx x$ or $\top$ in $\overline{A}$, the formula is deleted from S. If $\overline{A}$ contains $\bot$ or $x \not\approx x$, it can be removed from $\overline{A}$. If $\overline{A}$ contains a negative equality $x_1 \not\approx x_2$ with $x_1 \neq x_2$, it can be substituted away. The result is

$$\Pi \overline{x} \backslash \{x_2\}\ (\ \bigoplus \overline{A}[x_1 := x_2]\ ) \backslash \{x_1 \not\approx x_2\}.$$

The procedure stops when either the formula is deleted, or all elements in A have form 1. In that case, the formula is geometric.

At this point, we have given the complete transformation from PCL to geometric formulas, so that we have established a complete theorem proving procedure.

Transformation Geo is only the simplest possible transformation. It is clear that the transformation can be improved by trying to simplify the formula, by trying to apply antiprenexing, and by other transformations. Figure 10 lists some possible simplifications that can be considered.

Since the factoring rules in Figure 9 may cause existential blow up, one can also try to replace subformulas. Subformula replacement is also needed for proving the correctness of anti-Skolemization, since every predicate $\overline{F}(x_1, \ldots, x_n, y)$ can be viewed as an abbreviation for $f(x_1, \ldots, x_n) \approx y$.

Figure 9: Rewriting into Geometric Formulas

$$
\begin{array}{lcl}
A \oplus (B \otimes C) & \Rightarrow & (A \oplus B) \otimes (A \oplus C) \\
(A \otimes B) \oplus C & \Rightarrow & (A \oplus C) \otimes (B \oplus C)
\end{array}
$$

$$
\begin{array}{lcl}
(\Pi x\ P[x]) \oplus A & \Rightarrow & \Pi x\ (P[x] \oplus A) \\
A \oplus \Pi x\ P[x] & \Rightarrow & \Pi x\ (A \oplus P[x]) \\
\Pi x\ (P[x] \otimes Q[x]) & \Rightarrow & (\Pi x\ P[x]) \otimes (\Pi x\ Q[x])
\end{array}
$$

Figure 10: Possible Simplification Rules

$$
\begin{array}{lcl}
\Pi x\ (P[x] \oplus A) & \Rightarrow & (\Pi x\ P[x]) \oplus A \\
\Pi x\ (A \oplus P[x]) & \Rightarrow & A \oplus \Pi x\ P[x] \\
\Sigma x\ (P[x] \otimes A) & \Rightarrow & (\Sigma x\ P[x]) \otimes A \\
\Sigma x\ (A \otimes P[x]) & \Rightarrow & A \otimes (\Sigma x\ P[x])
\end{array}
$$

$$
\Sigma x\ (P[x] \oplus Q[x]) \Rightarrow (\Sigma x\ P[x]) \oplus (\Sigma x\ Q[x])
$$

$$
\begin{array}{lcl@{\qquad\qquad}lcl}
\top \oplus A & \Rightarrow & \top & \top \otimes A & \Rightarrow & A \\
A \oplus \top & \Rightarrow & \top & A \otimes \top & \Rightarrow & A \\
\bot \oplus A & \Rightarrow & A & \bot \otimes A & \Rightarrow & \bot \\
A \oplus \bot & \Rightarrow & A & A \otimes \bot & \Rightarrow & \bot
\end{array}
$$

$$
\begin{array}{lcl@{\qquad\qquad}lcl}
x \approx x & \Rightarrow & \top & \Pi x\ \top & \Rightarrow & \top \\
x \not\approx x & \Rightarrow & \bot & \Sigma x\ \bot & \Rightarrow & \bot
\end{array}
$$

We will now give the correctness proofs of the transformations, where 'correct' means that the transformation preserves strong representation. Both anti-Skolemization and transformation Geo extend the signature by introducing new predicate symbols. Correctness can be proven in two ways: Either one shows that the predicates have a definition, and uses the fact that definitions are conservative, or one uses the fact that there exists a predicate with the desired property, after which exists-introduction can be applied backwards. Due to the eigenvariable condition, this will introduce the predicate as a new predicate into the sequent. We will follow the second approach. Since the new symbol is a predicate symbol, we need to introduce quantification over predicates, and a corresponding introduction rule. We also add a $\otimes$-elimination rule to the calculus. This has the advantage that multiple formulas can be combined into a single formula.

Definition 6.5. *We extend the set of formulas, that was first defined in Defition 2.1, and extended in Definitions 2.17 and 3.10 with a second order, existential, Kleene quantifier:*

- *If F is a formula and p is a predicate symbol that occurs in F only with a fixed arity n, then $\Sigma p\ F$ is also a formula.*

Formulas of form $\Sigma p\ F$ are interpreted as follows: First define $I' \sim_p I$ if for every symbol s that is not the predicate symbol p, we have $I'(s) = I(s)$. Next define $R_P = \{I'(P) \mid I' \sim_p I\}$. Finally, define $I(\Sigma p\ F)$ by selecting from R_F the most preferred value, using the preference list for Σ in Figure 5.
We extend the calculus Seq_{PCL} *with the following deduction rules:*

$$\frac{S,\ P[q] \vdash \bot}{S,\ \Sigma p\ P[p] \vdash \bot} \qquad \frac{S,\ A \otimes B,\ S \vdash \bot}{S,\ A,\ B \vdash \bot}$$

In the first rule, predicate q must not occur in S or $P[p]$, and it must have the same arity as p.

It easy to prove for both rules that they preserve strong representation. The first rule is the natural extension of Σ-introduction to second order. The second rule tells that there is no need to distinguish between A and B separately, and $A \otimes B$ combined in a Kleene conjunction.

We now prove that positive subformula replacement is widening. (See Definition 2.15) Note that we do not assume that the formula is radicalized.

Theorem 6.6. *Let A be a formula that may contain PCL and Kleene operators. Let B be a subformula of A that is in the scope of $\Pi, \Sigma, \oplus, \otimes$, but no other operators. Assume that $x_1, \ldots, x_n$ are the free variables of B. Let p be a predicate symbol of arity n that does not occur in A. Then*

$$A[\ B[x_1, \ldots, x_n]\ ] \preceq \Sigma p \left\{ \begin{array}{l} \Pi x_1 \cdots x_n\ p_{\mathbf{f},\mathbf{e}}(x_1, \ldots, x_n) \oplus B[x_1, \ldots, x_n] \otimes \\ A[\ p_{\mathbf{t}}(x_1, \ldots, x_n)\ ] \end{array} \right.$$

Proof. Throughout the proof, we will abbreviate $x_1, \ldots, x_n$ as $\overline{x}$.

First assume that $I(A[B[\overline{x}]]) = \mathbf{f}$. Let I' be an interpretation for which $I' \sim_p I$. (which is equal to I except for the interpretation of predicate p) We have to show that $I'(\ (\ \Pi\overline{x}\ p_{\mathbf{f},\mathbf{e}}(\overline{x}) \oplus B[\overline{x}]) \otimes A[p_{\mathbf{t}}(\overline{x})]\ ) = \mathbf{f}$.

If there exists an interpretation J, with $J \approx_{\overline{x}} I',\ J(B[\overline{x}]) = \mathbf{f}$, and $J(p_{\mathbf{t}}(\overline{x})) \neq \mathbf{f}$, then we have $J(p_{\mathbf{f},\mathbf{e}}(\overline{x})) = \mathbf{f}$. This implies that $J(p_{\mathbf{f},\mathbf{e}}(\overline{x}) \oplus B[\overline{x}]) = \mathbf{f}$, which in turn implies that $I'(\Pi\overline{x}\ p_{\mathbf{f},\mathbf{e}}(\overline{x}) \oplus B[\overline{x}]) = \mathbf{f}$, which would complete the proof.

So we can assume that $J \sim_{\overline{x}} I'$ and $J(B[\overline{x}]) = \mathbf{f}$ implies $J(p_{\mathbf{t}}(\overline{x}) = \mathbf{f}$ in the rest of the proof. Once we have established this fact, it is easy to prove by induction on the number of operators in $A'[\ ]$, (which can be only $\Pi, \Sigma, \otimes$ or $\oplus$) that

$$\text{for all}\ \ J \sim_{\overline{x}} I' \text{ with } J(A'[B[\overline{x}]]) = \mathbf{f},\ \text{we have } J(A'[p_{\mathbf{t}}(\overline{x})]) = \mathbf{f}.$$

Since this applies to $A'[\ ] = A[\ ]$ and $J = I'$, the proof for $\mathbf{f}$ is complete.

Next assume that $I(A[B[\overline{x}]]) = \mathbf{t}$. Let F be the function from D^n to $\{\mathbf{f}, \mathbf{e}, \mathbf{t}\}$, that is defined by

$$F(d_1, \ldots, d_n) = I_{d_1}^{x_1} \cdots_{d_n}^{x_n} (B[x_1, \ldots, x_n]).$$

Define $I' = I_F^p$. It is clear that $I' \sim_p I$, and in every interpretation J with $J \sim_{\overline{x}} I'$, we have $J(B[\overline{x}]) = J(p(\overline{x}))$. It follows immediately that $I'(\ \Pi\overline{x}\ p_{\mathbf{f},\mathbf{e}}(\overline{x}) \oplus B[\overline{x}]\ ) = \mathbf{t}$. It remains to show (using induction on the number of operators in context $A'[\ ]$, which can be only $\Pi, \Sigma, \otimes, \oplus$), that for all interpretations J,

$$J \sim_{\overline{x}} I' \text{ and } J(A'[B[\overline{x}]]) = \mathbf{t} \text{ implies } J(A'[p_{\mathbf{t}}(\overline{x}]]) = \mathbf{t}.$$

This completes the proof. □

After having shown correctness of subformula replacement, which is used as part of the transformation Geo, but which also can be used separately as optimization, it remains to prove correctness of anti-Skolemization. It is straightforward to prove that truth of a formula implies truth of its anti-Skolemization. The introduced predicates $\overline{F}(\overline{x}, y)$ can be viewed as abbreviations for $f(\overline{x}) \approx y$. In order to show that falsehood is preserved as well, there are three possible ways to proceed: Firstly, one can apply Skolemization on the seriality axioms $\Pi\overline{x}\ \Sigma y\ \overline{F}_{\mathbf{t}}(\overline{x}, y)$, which will reintroduce the function symbols. This method was used in [9]. Secondly, one can use the fact that algorithm S_t only constructs interpretations (E, M) in which the interpetations of the predicates $\overline{F}$ are functional. This is because it introduces a witness only when it has to, and the only position where a predicate $\overline{F}$ can be false but not in conflict with an interpretation (E, M), is in the seriality axioms. Thirdly, one can give a direct proof. Because the only positive occurrence of a predicate $\overline{F}$ is in the seriality axioms $\Pi\overline{x}\ \Sigma y\ F_{\mathbf{t}}(\overline{x}, y)$, the interpretations of F can be made functional, without making any of the other formulas false. We will use the last approach.

Theorem 6.7. *Anti-Skolemization is a widening operation: Let A be a Kleene formula that is in NNF and radicalized. Let $f_1, \ldots, f_k$ be the function symbols*

that occur in A. Let n_j be the arity of f_j. Let A' be be obtained from A by anti-Skolemization. Let $\overline{F}_1, \ldots, \overline{F}_k$ be the predicate symbols that were introduced during anti-Skolemization. Then

$$A \preceq \Sigma \overline{F}_1 \cdots \overline{F}_k \bigotimes \left\{ \begin{array}{l} \Pi x_1 \cdots x_{n_1} \ \Sigma y \ \overline{F}_{1,\mathbf{t}}(x_1, \ldots, x_{n_1}, y) \\ \cdots \\ \Pi x_1 \cdots x_{n_k} \ \Sigma y \ \overline{F}_{k,\mathbf{t}}(x_1, \ldots, x_{n_k}, y) \\ A' \end{array} \right.$$

Proof. First observe that, due to the fact that A is radicalized, we have $I(A) \in \{\mathbf{f}, \mathbf{t}\}$, so that we could have used $\equiv$ instead of $\preceq$. Assume that $I(A) = \mathbf{t}$. We can extend I by setting

$$I(\overline{F_j})(d_1, \ldots, d_{n_j}, e) = \left\{ \begin{array}{ll} \mathbf{t} & \text{if } I(f_j)(d_1, \ldots, d_{n_j}) = e \\ \mathbf{f} & \text{otherwise} \end{array} \right.$$

It is clear that $I' \sim_{\overline{F}_1 \cdots \overline{F}_k} I$. The formula A' is obtained from A by a sequence of replacements $A_1, A_2, \ldots, A_m$, with $A_1 = A$, $A_m = A'$, and each replacement made in accordance to Definition 6.2. It follows from Lemma 6.8 that for each of the replacements we have $I'(A_{i+1}) = I'(A_i)$, so that it follows that $I(A') = \mathbf{t}$.

Next assume that there exists an interpretation I, s.t the right hand side of $\preceq$ is true in I. By the semantics of Σ, there exists an interpretation I', s.t. $I' \sim_{\overline{F}_1 \cdots \overline{F}_k} I$, $I'(A') = \mathbf{t}$, and for each j,

$$I'(\ \Pi x_1 \cdots x_{n_j} \ \Sigma y \ \overline{F}_{j,\mathbf{t}}(x_1, \ldots, x_{n_j}, y)\ ) = \mathbf{t}.$$

Using axiom of choice, it is possible to restrict the relations $\overline{F}$ in such a way that the derived relations $\overline{F}_t$ become functional.

Let I'' be an interpretation obtained in this way. We can apply Lemma 6.9, from which it follows that $I''(A') = \mathbf{t}$. Since the derived relations $\overline{F}_{j,\mathbf{t}}$ are functional, it is possible to extend I'' into an interpretation I''', where

$$I'''(f_j)(d_1, \ldots, d_{n_j}) = e \text{ iff } I'''(\overline{F}_{j,\mathbf{t}})(d_1, \ldots, d_{n_j}, e) = \mathbf{t}.$$

Now the replacements of Definition 6.2 can be made backwards until A is obtained. It follows from Lemma 6.8 that $I'''(A) = \mathbf{f}$. □

Lemma 6.8. *Let I be an interpretation with domain D. Let f be a function symbol with arity n. Let F be a predicate symbol with arity $n+1$. Assume that for all $d_1, \ldots, d_n, e \in D$, we have*

$$I(F)(d_1, \ldots, d_n, e) = \left\{ \begin{array}{ll} \mathbf{t} & \textit{if } I(f)(d_1, \ldots, d_n) = e \\ \mathbf{f} & \textit{otherwise} \end{array} \right.$$

Let $A[B]$ be a formula with a subformula B, that occurs only in the scope of operators $\Pi, \Sigma, \otimes, \oplus$. Assume that B contains a term $f(t_1, \ldots, t_n)$, and that no variable of $f(t_1, \ldots, t_n)$ is bound in B. Then

$$I(\ A[B[f(t_1, \ldots, t_n)]]\ ) = I(\ A[\ \Pi z \ F_{\mathbf{f},\mathbf{e}}(t_1, \ldots, t_n, z) \oplus B[z]\ ]\ ).$$

Lemma 6.8 tells for example that $I(\ \Pi x\ P(x, s(x))\ ) = I(\ \Pi xz\ S_{\mathbf{f},\mathbf{e}}(x, z) \oplus P(x, z)\ )$ if I interprets S as the relation that represents function s.

Lemma 6.9. *Let A be a Kleene formula that is in NNF and radicalized. Assume that A has form $(\ \Pi x_1 \cdots x_n\ \Sigma y\ F_{\mathbf{t}}(x_1, \ldots, x_n, y)\ ) \otimes A'$, and that all occurrences of predicate F in A' are of form $F_\lambda(t_1, \ldots, t_n, u)$ where λ does not contain $\mathbf{t}$. Let I be an arbitrary interpretation with domain D. Let I' be obtained from I by restricting $F_{\mathbf{t}}$ in such a way that the following conditions are met:*

- *$I' \sim_F I$,*
- *For all $d_1, \ldots, d_n, e \in D$, if $I(F)(d_1, \ldots, d_n, e) \neq \mathbf{t}$, then $I'(F)(d_1, \ldots, d_n, e) = I(F)(d_1, \ldots, d_n, e)$.*
- *For all $d_1, \ldots, d_n \in D$ there still exists an $e \in D$, s.t. $I(F)(d_1, \ldots, d_n, e) = \mathbf{t}$.*

Then $I(A') = \mathbf{t}$ implies that $I(A) = \mathbf{t}$.

7 Conclusions and Future Work

We have developed a sound and complete theorem proving procedure for Partial Classical Logic (PCL). A context is first decomposed into a set of sequents that represents the type assumptions and theorems that occur in it. After that, the resulting sequents are transformed into Kleene logic. Due to redundancy that is introduced by the decomposition into sequents, it is possible to ignore the possibility that formulas are ill-typed (are interpreted as $\mathbf{e}$) during further transformations on the sequents. This redundancy can be used to radicalize the Kleene formulas into formulas that always have a definite truth value. Once the formulas have been radicalized, various theorem proving strategies can be applied on them. In most cases, including all examples in this paper, the radicalizations of theorems can be treated as classical. We interpret this as: Once a theorem has been type checked, it can be treated as if it is classical.

The redundancy, introduced by decomposition of a context into sequents, made it possible to simplify the sequent calculus Seq^2_{PCL} of [8] into a new calculus $\mathrm{Seq}^{\preceq}_{PCL}$, which has fewer, and simpler reasoning rules. Because it is based on intermediate transformation to Kleene logic, it is closely related to the transformations that we have developed in this paper for theorem proving. Because of this, we included a description of $\mathrm{Seq}^{\preceq}_{PCL}$ in this paper.

In the near future, we will extend Geo ([10]), which currently uses untyped classical logic, to PCL. It follows from Theorem 2.14 that the same theorem proving procedure can be used for checking type conditions and for proving theorems. In practice, it will probably be necessary to use different settings. Type correctness proofs usually do not require equality reasoning, and reason on subformulas of a given goal only. We plan to study adaptions of the general procedure, that may be more effective.

We have chosen geometric logic as a starting point for theorem proving in PCL, because we expect that it is well-suited to problems where type conditions mix with other predicates, and that it will have good termination behaviour in case when a goal is not provable. These were the original reasons why geometric logic with flattening was introduced in [9]. In CASC 2007 ([20], [19]), Geo came second in the model finding category. Nevertheless, it is likely that superposition-based strategies can be used as well. In order to test this, the superposition calculus of [1] has to be adopted to Kleene logic.

8 Funding

This work was supported by the Polish National Science Center (Narodowe Centrum Nauki) [grant number DEC-2011/03/B/ST6/00346].

References

[1] Leo Bachmair and Harald Ganzinger. Rewrite-based equational theorem proving with selection and simplification. *Journal of Logic and Computation*, 4(3):217–247, 1994.

[2] Peter Baumgartner, Alexander Fuchs, Hans de Nivelle, and Cesare Tinelli. Computing finite models by reduction to function-free clause logic. *Journal of Applied Logic*, 7(1):58–74, 2009.

[3] Peter Baumgartner, Ulrich Furbach, and Ilkka Niemelä. Hyper Tableaux. In *Logics in Artificial Intelligence (JELIA '96)*, number 1126 in LNAI. Springer, 1996.

[4] Sergey Berezin, Clark Barrett, Igor Shikanian, Marsha Chechik, Arie Gurfinkel, and David Dill. 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)*, volume 125 of *Electronic Notes in Theoretical Computer Science*, pages 13–23. Elsevier, July 2005.

[5] Marc Bezem and Thierry Coquand. Automating coherent logic. In Geoff Sutcliffe and Andrei Voronkov, editors, *LPAR*, volume 3835 of *LNCS*, pages 246–260. Springer, 2005.

[6] François Bry and Sunna Torge. A deduction method complete for refutation and finite satisfiability. In Jürgen Dix, Luis Fariñas del Cerro, and Ulrich Furbach, editors, *JELIA*, volume 1489 of *LNCS*, pages 122–138. Springer Verlag, 1998.

[7] Ádám Darvas, Farhad Mehta, and Arsenii Rudich. Efficient well-definedness checking. In Alessandro Armado, Peter Baumgartner, and

Gilles Dowek, editors, *International Joint Conference on Automated Reasoning (IJCAR) 2008*, volume 5195 of *LNAI*, pages 100–115. Springer Verlag, 2008.

[8] Hans de Nivelle. Classical logic with partial functions. *Journal of Automated Reasoning*, 47(4):399–425, 2011.

[9] Hans de Nivelle and Jia Meng. Geometric resolution: A proof procedure based on finite model search. In John Harrison, Ulrich Furbach, and Natarajan Shankar, editors, *International Joint Conference on Automated Reasoning 2006*, volume 4130 of *Lecture Notes in Artificial Intelligence*, pages 303–317, Seattle, USA, August 2006. Springer.

[10] Hans de Nivelle and Jia Meng. theorem prover Geo 2007f. Can be obtained from my homepage, September 2006.

[11] Niklas Eén and Niklas Sörensson. An extensible SAT-solver [version 1.2], 2003.

[12] William Farmer. Andrews' type system with undefinedness. In C. Benzmüller, C. Brown, J. Siekmann, and R. Statman, editors, *Festschrift in the honour of Peter Andrews on his 70th birthday*, 2009. forthcoming.

[13] William M. Farmer and Joshua D. Guttman. A set theory with support for partial functions. *Studia Logica*, 66:59–78, 2000.

[14] Manfred Kerber and Michael Kohlhase. A mechanization of strong Kleene logic for partial functions. In *Automated Deduction - CADE 12*, volume 814 of *LNAI*, pages 371–385. Springer Verlag, June 1994.

[15] Sun Kim and Hantao Zhang. ModGen: Theorem proving by model generation. In Barbara Hayes-Roth and Richard Korf, editors, *Proceedings of AAAI-94*, pages 162–167, 1994.

[16] Joao Marques-Silva, Ines Lynce, and Sharad Malik. Conflict-driven clause learning SAT solvers. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability*, chapter 4, pages 131–153. IOS Press, 2009.

[17] Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. Chaff: Engineering an efficient SAT solver. In *Proceedings of the 38th annual Design Automation Conference*, pages 530–535. ACM, 2001.

[18] Andreas Nonnengart and Christoph Weidenbach. Computing small clause normal forms. In Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*, volume I, chapter 6, pages 335–367. Elsevier Science B.V., 2001.

[19] G. Sutcliffe. The CADE-21 Automated Theorem Proving System Competition. *AI Communications*, 21(1):71–82, 2008.

[20] G. Sutcliffe and C. Suttner. The State of CASC. *AI Communications*, 19(1):35–48, 2006.