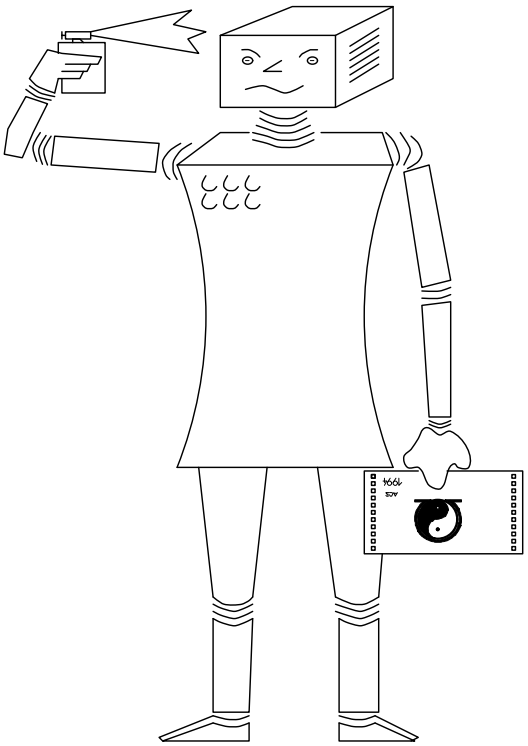
SEKI Report



Proving Ground Completeness of Resolution by Proof Planning

Manfred Kerber and Arthur C. Sehn

SEKI Report SR-96-05

Proving Ground Completeness of Resolution by Proof Planning

Manfred Kerber
The University of Birmingham
School of Computer Science
Birmingham, B15 2TT, England
e-mail: M.Kerber@cs.bham.ac.uk
http://www.cs.bham.ac.uk/~mmk

Arthur Christian Sehn*
Kaiserstr. 300
D-66386 St. Ingbert
Germany

Technical Report: CSRP-96-16
October 1996

Abstract

A lot of the human ability to prove hard mathematical theorems can be ascribed to a *problem-specific* problem solving know-how. Such knowledge is intrinsically incomplete. In order to prove related problems human mathematicians, however, can go beyond the acquired knowledge by adapting their know-how to new related problems. These two aspects, having rich experience *and* extending it by need, can be simulated in a proof planning framework: the problem-specific reasoning knowledge is represented in form of declarative planning operators, called methods; since these are declarative, they can be mechanically adapted to new situations by so-called meta-methods. In this contribution we apply this framework to two prominent proofs in theorem proving, first, we present methods for proving the ground completeness of binary resolution, which essentially correspond to key lemmata, and then, we show how these methods can be reused for the proof of the ground completeness of lock resolution.

1 Introduction

Machine-oriented theorem provers like those based on resolution, paramodulation, or rewriting have been successfully applied in different fields of logic and mathematics (see, e.g., [22, chapters 9,10]). The strength of these systems is remarkable, but the general

*Most of this work was done while I was working at the Universität des Saarlandes, FB Informatik, D-66041 Saarbrücken, Germany, e-mail: acsehn@cs.uni-sb.de.

complexity results demonstrate clearly that no algorithm can be constructed to practically solve arbitrary tasks, even propositional logic is in a class that is generally considered intractable. The success of human mathematicians can be largely ascribed to the fact that they are generally specialised in some fields and can rely on domain-specific problem solving techniques they have accumulated throughout their professional experiences. This kind of problem solving behaviour has been first supported in interactive theorem proving systems (e.g. LCF [11] or Nuprl [9]), which contain specific problem solving knowledge called tactics. In order to model the dynamic search process as well, tactics have been extended to so-called *methods* by adding specifications. Intuitively speaking, a method contains a piece of knowledge for solving or simplifying problems or transforming them into a form that is easier to solve. Specifications of methods essentially consist of pre- and postconditions for the method to allow connecting them to proof plans. These proof plans are the basic elements of a more general planning framework (e.g. CLAM [7, 8, 6]).

The reasoning power of such plan-based systems is not derived from a (theoretically) complete underlying reasoning calculus, but relies on domain-specific problem solving knowledge. Indeed the methods may be theoretically complete, practically this is of minor importance. What matters is, whether a problem solver, either human or computer, has the necessary domain-specific problem solving knowledge that offers the chance to solve the problem with a reasonable amount of search. Such domain-specific knowledge is, however, for non-trivial domains inevitably limited and hence practically incomplete. This leads to a limited reasoning power of the corresponding system. While this holds for plan-based systems and human mathematicians alike, the latter often go beyond their specialised knowledge. They have the ability to adapt existing methods to novel situations, which is one of the main features contributing to our problem solving competence (see [19] for mathematical reasoning and [21] for general problem solving). Although very important, this issue remains widely unaddressed in proof planning systems. Actually in a framework where tactics are described purely procedurally, this task is quite formidable and equals to the problem of mechanically modifying procedures.

In previous work, we have developed a declarative approach for representing methods that enables a mechanical modification and adaptation to novel tasks [15, 16]. In the following section we shortly recall the general framework (the description of the formal framework as well as parts of this introduction are based on [13]) and then present the user-written methods for proving the ground completeness of binary resolution. In doing so we follow Bledsoe's formalisation of the problem [3], in particular we use the lemmata proposed there for working out the main theorem. This proof is the most advanced case study that has been done with the Ω-MKRP proof development environment [14] up to now and is first described in [20]. In this work we show that our general framework can not only be applied to specify and automatically solve this problem, but that this proof can even be transferred to a similar problem, namely the ground completeness of lock resolution. The general idea of using the first proof as a guideline for the second has been investigated by Bledsoe [3] too. We discuss the differences between the two approaches in Section 3, where we present the two examples in detail. Before doing so we recapitulate in the next section our proof planning framework.

2 A Declarative Approach toward Proof Planning

In this section we present our approach to proof planning and show how methods can be declaratively represented and mechanically modified.

The Planning Framework

The work in this paper should be understood in the setting of a computational model that casts the entire process of theorem proving, from the analysis of a problem up to the completion of a proof, as an *interleaving process* of proof planning, method execution and verification. In particular, this model ascribes a problem solver's reasoning competence to the existence of methods together with a planning mechanism that uses these methods for proof planning.

To understand the proof planning process, please remember that the goal of proof planning is to fill gaps in a given partial proof tree by forward and backward reasoning [15]. Thus from an abstract point of view the planning process is the process of exploring the search space of *planning states* generated by the *plan operators* in order to find a complete *plan* (i.e. a sequence of instantiated plan operators) from a given *initial state* to a *terminal state*.

Concretely a *planning state* contains a subset of proof lines which are formulated in Gentzen's natural deduction (**ND** for short) calculus in the current partial proof that corresponds to the boundaries of a gap in the proof—note that we adopted a linearised version of ND proofs as introduced in [2]. This subset can be divided into *open lines* (that must be proved to bridge the gap) and *support lines* (that can be used as premises to bridge it). The initial planning state consists of all lines in the initial problem; the assumptions are the support lines and the conclusion is the only open line. The terminal planning state is reached when there is no more open line in the planning state.

A Declarative Representation for Methods

In order to have the possibility to adapt methods mechanically, we represent methods in a mainly declarative manner, only an almost constant "procedural content" is non-declarative. Formally, a method in our approach is defined as a 6-tuple with the components:

Declarations: A signature that declares the meta-variables used in the method,

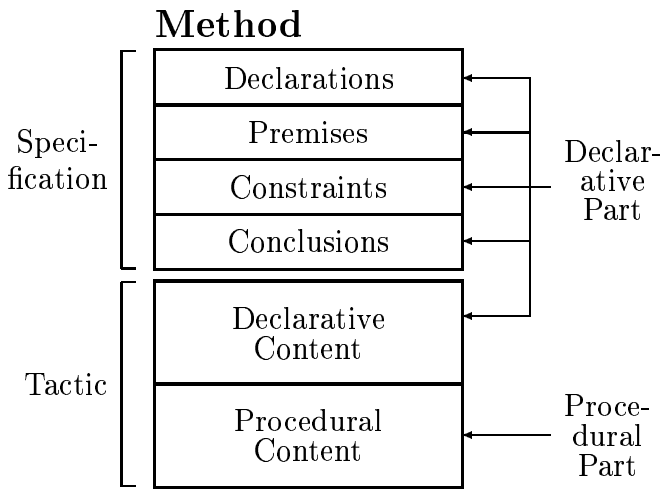
Premises: Schemata of proof lines which are used by this method to prove the conclusions,

Constraints: Additional restrictions on the premises and the conclusions, which can not be formulated in terms of proof line schemata (for a detailed description of the constraint language see [16]),

Conclusions: Schemata of proof lines which this method is designed to prove,

Declarative Content: A piece of declarative knowledge interpreted by the procedural content. This slot is currently restricted to schemata of partial proofs,

Procedural Content: Either a standard procedure interpreting the declarative content, or a special purpose inference procedure.



Methods can be very general, for instance consist essentially of one application of a rule at the calculus level. More specific methods comprise proof ideas, for instance, to use a homomorphy property, to apply mathematical induction or to use diagonalisation. Such methods are typically formulated in terms of proof schemata. The most specific methods consist of full proofs for specific problems. In order to clarify this notion, let us look at a trivial example. It is derived from a calculus level rule, namely implication introduction.

Method : `implies-i`	
Declarations	L_1, L_2, L_3 : prln; H_2, H_3 : list(prln); Ψ, Φ : term;
Premisses	$\oplus L_2$
Constraint	$H_2 \leftarrow \mathsf{listcons}(L_1, H_3)$
Conclusions	$\ominus L_3, \oplus L_1$
Declarative Content	$L_1.\quad (L_1) \vdash \Phi \qquad (\mathtt{Hyp})$ $L_2.\quad H_2 \vdash \Psi \qquad (OPEN)$ $L_3.\quad H_3 \vdash \Phi \rightarrow \Psi \qquad (\rightarrow I\ L_2)$
Procedural Content	`schema – interpreter`

Figure 1: The `implies-i` method

This method proposes to prove an implication that matches L_3 (Φ and Ψ are meta-variables that can match with arbitrary formulae, all meta-variables are declared in the "Declarations" slot.) by introducing a hypothesis (method line L_1) and the succedent formula as a new goal in method line L_2. L_1 and L_2 are newly introduced into the current planning state by this method; this is indicated by the label $\oplus$. In the same way we label L_3 with a $\ominus$, since it can be deleted from the planning state. In addition to the existence of L_3 in the current planning state, the constraint of this method has to be fulfilled, what is trivial in this case, since the constraint just binds the assumption list of line L_2 (bound to the meta-variable H_2) to the result of appending the line L_1 to the assumption list of the line L_3 (for a detailed description of the constraint language see [16]).

Planning with Methods

The key feature of our method language is that the planning operator is directly derived from the specification of a method. However, the specification only gives a static view (viewed from the completed proof) of the method which is inadequate for describing the dynamic behaviour needed in proof planning. Statically a method derives its conclusions from its premises. Dynamically, it is important to declare which lines in the specification have to be present in the planning state for the method to be applicable (we call them *required* lines), and which are constructed by the method. The first are lines without an annotation and they will not be modified by the method in which they occur (e.g. lines with a definition or a lemma). The latter are lines annotated by the label "$\oplus$" in the method specification. In contrast to these, we have lines that are no longer useful for the planner and which can be deleted from the current planning state. They are annotated by the label "$\ominus$".

From such an operator, it is easy to construct a STRIPS plan operator that has three slots: the precondition list, the delete list, and the add list.

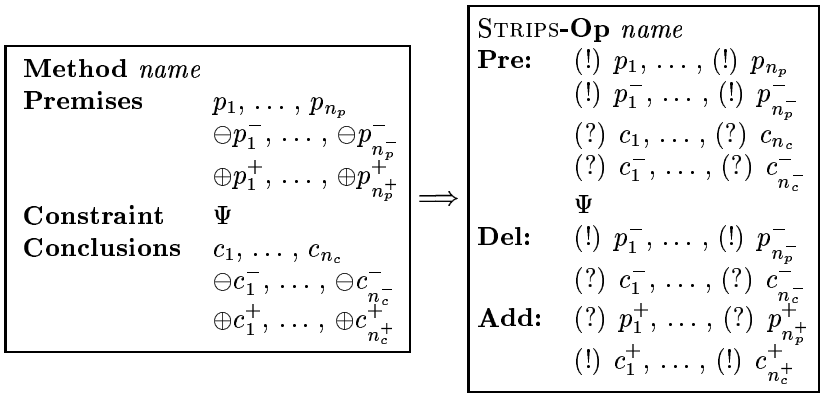


Figure 2: Generation of STRIPS operators from methods

While in Figure 2 the most general case of such a translation is represented, usually

not all possible labelings are used in a single method. The unlabelled lines in a method go to the preconditions slot and the lines with a "$\oplus$" to the add slot. The lines labelled with "$\ominus$" are moved to the precondition slot and to the delete slot. In the preconditions and the delete slot the premises become support lines and the conclusions become open lines. In the add slot it is the other way round. The content of the constraint slot Ψ is inserted into the preconditions slot of a STRIPS operator. Note that Ψ is not a proof line that is present in the planning state, but a formula specifying an additional applicability condition.

For our case study we essentially use a variant of the following planning algorithm:

While there are still open lines in the planning state

1. Find all possibilities of an application of the methods
   (a) Select a method M.
   (b) Find all possibilities of matching the lines in M's precondition slot with lines in the planning state.
   (c) Evaluate the *constraint* of M with the bindings established in 1(b).
2. Select the "best" method $\widetilde{M}$ (this is the choice point for the backtracking mechanism and the point where some heuristic control can take place)
3. Apply the plan operator $\widetilde{M}$ to the planning state
   (a) Insert the lines in the add slot of $\widetilde{M}$ into the planning state.
   (b) Remove the lines in the delete slot of $\widetilde{M}$ from the planning state.

Note that backward planning is not possible from the viewpoint of this algorithm which uses the translated STRIPS-operators since the terminal state is defined by the absence of open lines. However, in the static model, that is, in the space of proof lines, we can model by this planning approach reasoning forwards from the assumptions towards the open lines as well as reasoning backwards from the open lines towards the support lines. During the matching of the lines in the preconditions slot and the evaluation of the constraint all meta-variables should have been bound to object level entities. Therefore, the new lines of step 3.(a) can be constructed by simply instantiating the meta-variables.

Once a complete proof plan is found, all methods (i.e. their tactics) in the proof plan are successively executed in order to construct a calculus level proof. The verification phase, which follows the application of the methods, may result in a recursive call to the planner or in backtracking. While a recursive call refines a plan and models hierarchical planning, the backtracking rejects the plan and calls the proof planner in order to find an alternative one.

Modification of Methods

The standard practice of proof planning is to use a fixed repertoire of methods in order to produce a proof plan [6]. Difficulties arise when new problems are encountered exceeding the power of the existing methods. To overcome this, a main feature contributing to the problem solving competence of mathematicians becomes crucial, namely that they can extend their current problem solving repertoire by adapting existing methods to suit novel situations.

By adopting a declarative approach for formulating methods, it is firstly possible to extract methods from proofs, secondly it is also feasible to formulate meta-methods adapting existing methods. Currently, a meta-method is essentially a procedure which takes as input some methods and some additional parameters, and produces a new method. We have already identified a variety of meta-methods such as: the generalisation of methods in order to apply them in less specific situations or the syntactic adaptation of methods to bridge syntactic gaps, for instance, arities of predicates.

One potential criticism is that we should instead construct more general methods which cover large classes of problems. Although general methods are definitely needed for effective proof planning systems, this by no means excludes the need of modification. It is very difficult, for instance, to come up with a single method covering all possible cases which a human mathematician would *intuitively* consider as an instance of homomorphy. A well-known example is the rippling method developed in Bundy's group, which has been extended from *rippling-out* [5] to a method covering a wide range of related problems [6].

The intention of our work can be compared to Ireland's approach of proof critics [17]. While proof critics are specific and attached to single methods, meta-methods embody general problem independent procedures for adapting arbitrary methods which meet some applicability conditions. The related work of Giunchiglia and Traverso [10] to represent tactics in a logical meta-language has a similar motivation as well, namely to mechanically adapt existing tactics. Their formalism is more expressive since they can represent procedural aspects like loops on a logical meta-level too. In our approach, the declarative part of methods basically consists of a proof schema. This leads to a more natural representation and enables an easier transformation in some cases.

3 Two Completeness Proofs as Case Study

In this section we want to illustrate our proof planning framework by the example of proving the refutation completeness of binary and locking ground resolution. First, a proof plan for the completeness of the binary resolution is constructed, and then this proof plan is reused and adapted to a proof plan for the completeness of lock.

In order to give a formal proof of this theorem we follow Bledsoe's encoding [3]. In this work, Bledsoe describes how a proof of the completeness of lock resolution can be derived by analogy from a given proof plan for binary resolution. Although we use the same logical encoding, there are some differences in the approaches. Firstly, our notion of a proof plan

is different from Bledsoe's: In his framework a proof plan is a list of valid logical theorems, whereas in our framework a proof plan is a list of plan operators, the methods. Secondly, Bledsoe assumes a proof plan for binary resolution to be *given* (as Bledsoe we assume the lemmata to be given and don't dig into their proofs), we, however, generate this plan by a semi-automated planning process. Thirdly, Bledsoe uses *abduction* to derive the analogous proof plan for lock resolution, while we use procedures, the so-called *meta-methods*, to adapt the planning operators in the proof plan for binary resolution completeness to a proof plan for lock resolution.

The Completeness Proof of Ground Resolution

The refutation completeness of ground resolution is well known and can be stated as follows (cf. [18]):

Theorem: *If S is a finite, unsatisfiable set of ground clauses, then there is a derivation of the empty clause $\Box$ by applying resolution to S, i.e., $\exists n.\Box \in Res(S,n)$, where $Res(S,n)$ is inductively defined for all $n \in \mathbb{N}$ as*

$$Res(S,n) = \begin{cases} S, \textit{ if } n = 0, \\ \bigcup\limits_{i,j<n} \{R \mid R \textit{ is resolvent of } C_1 \textit{ \& } C_2, C_1 \in Res(S,i), C_2 \in Res(S,j)\}, \textit{ else} \end{cases}$$

The proof of this theorem is an inductive argument on the k–parameter $exl(S)$ (also called excess literal number) of a set of clauses S which is defined as the number of occurrences of literals in every clause of S minus the number of clauses in S. More formally, it can be written as: $exl(S) = [\sum_{C \in S} |C|] - |S|$, where $|C|$ is the number of literals in the clause C and $|S|$ is the number of clauses in the clause set S (this so-called k–parameter technique was introduced in [1]).

We now turn to the proof of the completeness of binary ground resolution. The basic idea is to use an inductive argument on the excess literal number of the clause set S. We do not describe the base case[1], but we will concentrate on finding a proof plan for the inductive step.

The initial planning state is displayed in Figure 3. The lemmata LEM0 through LEM8 are exactly the lemmata given in Bledsoe's paper [3]. In addition, we need in our formulation lemmata for dealing with sets, lemmata SUBS-S1-S and SUBS-S2-S are two subsumption properties of two special clause sets (see [20] for detailed proof plans for them); the other hypotheses are standard definitions and properties of set theory. In Bledsoe's approach these hypotheses are not necessary, since they are basic to the Str+ve-Prover [12] for set theory. These properties are easy properties of the subset relation and can straightforwardly be proved either by calling an automated theorem prover or by further planning processes. The support lines are marked with "(!)", the only open line is marked with

[1] In the case $exl(S) < 0$, $\Box$ must be in $S = Res(S,0)$. If we have $exl(S) = 0$, but $\Box \notin S$, the set of clauses S consists entirely of unit clauses. Since a set of unit clauses is only unsatisfiable, if it contains two complementary units, the empty clause can derived by an application of the resolution rule to a pair of such units, that is, $\Box \in Res(S,1)$.

(!) SUBS-S1-S.	SUBS-S1-S	$\vdash \forall S.\forall C.\forall L.subsumes(S \setminus\{C\} \cup \{C \setminus_{cl}\{L\}\}, S)$	(Hyp)
(!) SUBS-S2-S.	SUBS-S2-S	$\vdash \forall S.\forall C.\forall L.[L \in C \rightarrow$ $subsumes(S \setminus\{C\} \cup \{\Box \cup_{cl} \{L\}\}, S)]$	(Hyp)
(!) LEM-13.	LEM-13	$\vdash \forall S.\forall S'.\forall C.[[C \in S \wedge S' \subseteq S] \rightarrow S' \cup \{C\} \subseteq S]$	(Hyp)
(!) LEM-14.	LEM-14	$\vdash \forall S.\forall S'.\forall C.[[C \in S \wedge S \subseteq S'] \rightarrow C \in S']$	(Hyp)
(!) $\subseteq$-Trans.	$\subseteq$-Trans	$\vdash \forall S_1.\forall S_2.\forall S_3.[[S_1 \subseteq S_2 \wedge S_2 \subseteq S_3] \rightarrow S_1 \subseteq S_3]$	(Hyp)
(!) $\cup$-Def.	$\cup$-Def	$\vdash \forall S.\forall C.\forall C'.[C' \in S \cup \{C\} \leftrightarrow [C' \in S \vee C' = C]]$	(Hyp)
(!) $\setminus$-Def.	$\setminus$-Def	$\vdash \forall S.\forall C.\forall C'.[C' \in S \setminus\{C\} \leftrightarrow [C' \in S \wedge \neg[C' = C]]]$	(Hyp)
(!) $\cup_{cl}$-Def.	$\cup_{cl}$-Def	$\vdash \forall C.\forall L.\forall L'.[L' \in C \cup_{cl} \{L\} \leftrightarrow [L' \in C \vee L' = L]]$	(Hyp)
(!) $\setminus_{cl}$-Def.	$\setminus_{cl}$-Def	$\vdash \forall C.\forall L.\forall L'.[L' \in C \setminus_{cl}\{L\} \leftrightarrow$ $[L' \in C \wedge \neg[L' = L]]]$	(Hyp)
(!) $\subset$-Prop1.	$\subset$-Prop1	$\vdash \forall C.\forall L.[L \in C \rightarrow C \setminus_{cl}\{L\} \subseteq C]$	(Hyp)
(!) $\subset$-Prop2.	$\subset$-Prop2	$\vdash \forall S.\forall C.\forall L.S \setminus\{C\} \cup \{C \setminus_{cl}\{L\} \cup_{cl} \{L\}\} = S$	(Hyp)
(!) $\subset$-Prop3.	$\subset$-Prop3	$\vdash \forall C.\forall L.[[nonunit(C) \wedge L \in C] \rightarrow \Box \cup_{cl} \{L\} \subseteq C]$	(Hyp)
(!) $\subseteq$-Def.	$\subseteq$-Def	$\vdash \forall S_1.\forall S_2.[S_1 \subseteq S_2 \leftrightarrow \forall C.[C \in S_1 \rightarrow C \in S_2]]$	(Hyp)
(!) $\subseteq$-Def.	$\subseteq$-Def	$\vdash \forall C_1.\forall C_2.[C_1 \subseteq C_2 \leftrightarrow \forall L.[L \in C_1 \rightarrow L \in C_2]]$	(Hyp)
(!) LEM0.	LEM0	$\vdash \forall S.\forall S'.[[finite(S) \wedge [unsat(S) \wedge$ $subsumes(S', S)]] \rightarrow [finite(S') \wedge unsat(S')]]$	(Hyp)
(!) LEM1.	LEM1	$\vdash \forall S.\forall C.\forall L.\forall n.[\Box \in Res(S \cup \{C\}, n) \rightarrow$ $[\Box \in Res(S \cup \{C \cup_{cl} \{L\}\}, n) \vee$ $\Box \cup_{cl} \{L\} \in Res(S \cup \{C \cup_{cl} \{L\}\}, n)]]$	(Hyp)
(!) LEM2.	LEM2	$\vdash \forall S.[[\neg[\Box \in S] \wedge 0 < exl(S)] \rightarrow$ $\exists C.\exists L.[C \in S \wedge [nonunit(C) \wedge L \in C]]]$	(Hyp)
(!) LEM2A.	LEM2A	$\vdash \forall S.\forall C.\forall L.[[C \in S \wedge [nonunit(C) \wedge L \in C]] \rightarrow$ $\exists S_1.\exists S_2.\exists C_1.\exists C_2.[C_1 = C \setminus_{cl}\{L\} \wedge$ $[C_2 = \Box \cup_{cl} \{L\} \wedge [S_1 = S \setminus\{C\} \cup \{C_1\}$ $\wedge S_2 = S \setminus\{C\} \cup \{C_2\}]]]]$	(Hyp)
(!) LEM3.	LEM3	$\vdash \forall S.\forall C.\forall C'.[[C \in S \wedge C' \subseteq C] \rightarrow$ $exl(S \setminus\{C\} \cup \{C'\}) < exl(S)]$	(Hyp)
(!) LEM4.	LEM4	$\vdash \forall S.\forall n.S \subseteq Res(S, n)$	(Hyp)
(!) LEM5.	LEM5	$\vdash \forall S.\forall S'.\forall n.[S \subseteq S' \rightarrow Res(S, n) \subseteq Res(S', n)]$	(Hyp)
(!) LEM6.	LEM6	$\vdash \forall S.\forall n_1.\forall n_2.Res(Res(S, n_1), n_2) = Res(S, n_1 + n_2)$	(Hyp)
(!) LEM7.	LEM7	$\vdash \forall S.\forall S'.[\forall C.[C \in S \rightarrow$ $\exists C'.[C' \in S' \wedge C' \subseteq C]] \rightarrow subsumes(S', S)]$	(Hyp)
(!) LEM8.	LEM8	$\vdash \forall C.\forall C'.[[C \subseteq C' \wedge \neg[C' \subseteq C]] \rightarrow C \subseteq C']$	(Hyp)
(?) THM.	$\mathcal{H}_1$	$\vdash \forall S.[[finite(S) \wedge [unsat(S) \wedge [0 < exl(S) \wedge$ $[\neg[\Box \in S] \wedge \forall S'.[[finite(S') \wedge [unsat(S') \wedge$ $exl(S') < exl(S)]] \rightarrow \exists n'.\Box \in Res(S', n')]]]]]$ $\rightarrow \exists n.\Box \in Res(S, n)]$	(*OPEN*)

Figure 3: The initial planning state for the completeness of binary ground resolution

"(?)" which is the theorem to prove. The hypotheses of the theorem, abbreviated as $\mathcal{H}_1$, are all supporting lines, that is, all lines except the theorem itself. In addition to the initial planning state we have a set of methods which can be selected as planning operators within the planning process.

In the example of the resolution completeness proof, we can figure out different kinds of methods: The one category of methods corresponds to the usual **ND** inference rules (e.g., `forall-i`, `implies-i`, and `and-e`. For `implies-i` see Figure 1). The other category of methods are specialised methods for the application of the given lemmata[2]. One such method can be seen in Figure 4. When different methods are applicable the proof planner prefers the more specialised one of the second category over the general ones of the first category. Intuitively, the method has the following meaning: The lemma 2 (method line $LEM2$) is applied to the open line L_{13} which is replaced by the open line L_{11} in the current planning state. In addition, the supporting lines L_8 through L_{10} are inserted into the planning state representing the partitioning of the clause set S: Select a non-unit clause C in S and create two clause sets S_1 and S_2, where S_1 is built from S by replacing the non-unit clause C by deleting a literal L from it, and S_2 by replacing C by the unit clause with the single literal L. The key idea of the completeness proof is the application of the induction hypothesis to these two clause sets S_1 and S_2. The derivations of the empty clause resulting from S_1 and S_2 can be concatenated resulting in a derivation of the empty clause starting with S (the concatenation is correct, since we have no free variables in the derivations).

When applying the planning procedure to the initial state using the methods above, we get the following proof plan:

> ⟨`forall-i`, `implies-i`, `and-e`, `and-e`, `and-e`, `and-e`, `apply-lem2`, `apply-lem2A`, `apply-lem0`, `apply-lem3`, `apply-lem0`, `apply-lem3`, `apply-ind`, `apply-ind`, `apply-lem1`, `rewrite-set`, `apply-lem4`, `apply-lem13`, `apply-lem5`, `apply-lem6`, `apply-lem14`⟩

After executing this plan we obtain an **ND** proof for the original problem where five lines representing lemmata still have to be proved (this can be done by another call of the planner reflecting hierarchical planning or by calling an automated theorem prover). The full proof contains 94 steps and can be found in the appendix A.

The Completeness Proof of Ground Lock Resolution

Next we want to prove the completeness of lock resolution. Lock resolution is a refinement of binary resolution where an index referring to an ordering is assigned to each literal in a clause and only resolving on minimal literals respecting the ordering is allowed (cf. [4, 18]).

[2]Of course it would be very useful to have one single general method for the application of arbitrary definitions or lemmata. It is, however, a non-trivial problem to consider all possibilities, in particular, when applying an **ND** rule for quantified formulae, a special treatment for the hypotheses is needed. For instance, the eigenvariable condition must be met. Until these problems are solved, the task of generating methods from lemmata has mainly to be fulfilled by a human user.

Method : `apply-lem2`	
Declarations	$LEM2, L_1, L_2, L_3, L_4, L_5, L_6, L_7, L_8, L_9, L_{10}, L_{11}, L_{12}, L_{13} : \mathsf{prln}$; $H_1, H_2, H_3, H_4 : \mathsf{list}(\mathsf{prln})$; $J_1, J_2 : \mathsf{just}$; $S', S'', n, C'', L'' : \mathsf{var}$; $S, C, L : \mathsf{const}$
Premisses	$\ominus LEM2, \ominus L_2, \ominus L_3, \oplus L_{11}$
Constraint	$C \leftarrow \mathsf{newconst}(\mathsf{termtype}(C'')) \,\&\, L \leftarrow \mathsf{newconst}(\mathsf{termtype}(L'')) \,\&$ $H_3 \leftarrow \mathsf{listcons}(L_6, H_2) \,\&\, H_4 \leftarrow \mathsf{listcons}(L_7, H_3) \,\&$ $\bigwedge^{\mathsf{listcons}(LEM2, H_1)} X : \mathsf{prln.termoccs}(C, \mathsf{prlnformula}(X)) \doteq \langle\rangle \,\&$ $\bigwedge^{H_3} x : \mathsf{prln.termoccs}(L, \mathsf{prlnformula}(X)) \doteq \langle\rangle$
Conclusions	$\ominus L_{13}, \oplus L_1, \oplus L_4, \oplus L_5, \oplus L_6, \oplus L_7, \oplus L_8, \oplus L_9, \oplus L_{10}, \oplus L_{12}$
Declarative Content	$LEM2.$ $LEM2$ $\vdash \forall S''.[[\neg[\Box \in S''] \wedge 0 < \mathit{exl}(S'')] \rightarrow \exists C''.\exists L''.[C'' \in S'' \wedge [\mathit{nonunit}(C'') \wedge L'' \in C'']]]$ (Hyp) $L_1.$ $LEM2$ $\vdash [[\neg[\Box \in S] \wedge 0 < \mathit{exl}(S)] \rightarrow \exists C''.\exists L''.[C'' \in S \wedge [\mathit{nonunit}(C'') \wedge L'' \in C'']]]$ $(\forall E\ LEM2)$ $L_2.$ H_1 $\vdash 0 < \mathit{exl}(S)$ (J_1) $L_3.$ H_1 $\vdash \neg[\Box \in S]$ (J_2) $L_4.$ H_1 $\vdash [\neg[\Box \in S] \wedge 0 < \mathit{exl}(S)]$ $(\wedge I\ L_3, L_2)$ $L_5.$ H_1, $LEM2$ $\vdash \exists C''.\exists L''.[C'' \in S \wedge [\mathit{nonunit}(C'') \wedge L'' \in C'']]$ $(\rightarrow E\ L_4, L_1)$ $L_6.$ H_3 $\vdash \exists L''.[C \in S \wedge [\mathit{nonunit}(C) \wedge L'' \in C]]$ (Hyp) $L_7.$ H_4 $\vdash [C \in S \wedge [\mathit{nonunit}(C) \wedge L \in C]]$ (Hyp) $L_8.$ H_4 $\vdash C \in S$ $(\wedge E*\ L_7)$ $L_9.$ H_4 $\vdash \mathit{nonunit}(C)$ $(\wedge E*\ L_7)$ $L_{10}.$ H_4 $\vdash L \in C$ $(\wedge E*\ L_7)$ $L_{11}.$ H_4 $\vdash \exists n.\Box \in \mathit{Res}(S, n)$ (OPEN) $L_{12}.$ H_3 $\vdash \exists n.\Box \in \mathit{Res}(S, n)$ $(\exists E\ L_6, L_{11})$ $L_{13}.$ H_2 $\vdash \exists n.\Box \in \mathit{Res}(S, n)$ $(\exists E\ L_5, L_{12})$
Procedural Content	`schema – interpreter`

Figure 4: The `apply-lem2` method

The completeness proof also uses the k-parameter technique and is almost the same as in the case of binary resolution. The only exception is that one uses a *maximal* literal to partition the clause set S. Therefore an additional predicate $Q(S, L)$ has to be added to the lemmata 1, 2, and 2A which describes the maximality property of the selected literal. More formally, $Q(S, L)$ is an abbreviation of $\forall C'.\forall L'.[C' \in S \wedge nonunit(C') \rightarrow i(L') \leq i(L)]$, where i is the index of the given literal. Bledsoe's precondition prover returns a weaker definition of $Q(S, L)$ (cf. [3]) sufficient for the proof but this is less intuitive.

The initial planning state is just that displayed in Figure 3 where the constant *unsat* is replaced by *unsat-L* and *Res* to *Res-L* reflecting that unsatisfiability and the resolution rule have to be taken with respect to locking. The altered lemmata 1, 2, and 2A have the following form:

(!) LEM1. LEM1 $\vdash \forall S.\forall C.\forall L.\forall n.[[\square \in Res\text{-}L(S \cup \{C\}, n) \wedge Q(S, L)] \rightarrow [\square \in Res\text{-}L(S \cup \{C \cup_{cl} \{L\}\}, n) \vee \square \cup_{cl} \{L\} \in Res\text{-}L(S \cup \{C \cup_{cl} \{L\}\}, n)]]$ (Hyp)

(!) LEM2. LEM2 $\vdash \forall S.[[\neg[\square \in S] \wedge 0 < exl(S)] \rightarrow \exists C.\exists L.[C \in S \wedge [nonunit(C) \wedge [L \in C \wedge Q(S, L)]]]]$ (Hyp)

(!) LEM2A. LEM2A $\vdash \forall S.\forall C.\forall L.[[C \in S \wedge [nonunit(C) \wedge [L \in C \wedge Q(S, L)]]] \rightarrow \exists S_1.\exists S_2.\exists C_1.\exists C_2.[C_1 = C \setminus_{cl} \{L\} \wedge [C_2 = \square \cup_{cl} \{L\} \wedge [S_1 = S \setminus \{C\} \cup \{C_1\} \wedge S_2 = S \setminus \{C\} \cup \{C_2\}]]]]$ (Hyp)

Now we do not start a search for a new plan for proving the completeness of lock resolution, but we try to patch the already found proof plan for binary resolution. The adaptation of the original proof to lock resolution can be achieved by the following meta-methods:

- Map the constants *unsat* to *unsat-L* and *Res* to *Res-L* in all methods used in the proof plan,
- Add in the methods with the schematic lines for the lemmata 1, 2 and 2A the additional predicate $Q(S, L)$.

The first step can be done by an analogical syntactic mapping, whereas the second needs further refinement. Since the lemmata 1, 2, and 2A are applied in the corresponding methods `apply-lem1`, `apply-lem2`, and `apply-lem2A`, these methods must be patched as follows: For the first method, the `apply-lem1` method, the predicate $Q(S, L)$ must be added to the method lines where the preconditions of lemma 1 are used. For the second, the `apply-lem2`, we have altered the method lines stemming from `apply-lem2` to the lines which are all marked with a prime sign (cf. Figure 5). Additionally, a new schematic method line, namely line L''_{10}, was inserted into the declarative content of the method. Note, that this additional line is inserted as a supporting line into the planning state representing the fact that the predicate $Q(S, L)$ must hold, when this method is applied. For the third method, the `apply-lem2A` method, a similar adaptation takes place which

Method : `apply-lock-lem2`	
Declarations	$LEM2, L'_1, L_2, L_3, L_4, L'_5, L'_6, L'_7, L'_8, L'_9, L'_{10}, L''_{10}, L_{11}, L_{12}, L'_{13}$: prln; H_1, H_2, H_3, H_4 : list(prln); J_1, J_2 : just; S', S'', n, C'', L'' : var; S, C, L : const
Premisses	$\ominus LEM2, \ominus L_2, \ominus L_3, \oplus L_{11}$
Constraint	$C \leftarrow \mathsf{newconst}(\mathsf{termtype}(C'')) \;\&\; L \leftarrow \mathsf{newconst}(\mathsf{termtype}(L'')) \;\&$ $H_3 \leftarrow \mathsf{listcons}(L'_6, H_2) \;\&\; H_4 \leftarrow \mathsf{listcons}(L'_7, H_3) \;\&$ $\bigwedge^{\mathsf{listcons}(LEM2,H_1)} X : \mathsf{prln.termoccs}(C, \mathsf{prlnformula}(X)) \doteq \langle\rangle \;\&$ $\bigwedge^{H_3} x : \mathsf{prln.termoccs}(L, \mathsf{prlnformula}(X)) \doteq \langle\rangle$
Conclusions	$\ominus L'_{13}, \oplus L'_1, \oplus L_4, \oplus L'_5, \oplus L'_6, \oplus L'_7, \oplus L'_8, \oplus L'_9, \oplus L'_{10}, \oplus L''_{10}, \oplus L_{12}$
Declarative Content	$LEM2.$ $LEM2$ $\vdash \forall S''.[[\neg[\Box \in S''] \wedge 0 < exl(S'')] \rightarrow \exists C''.\exists L''.[C'' \in S'' \wedge [nonunit(C'') \wedge [L'' \in C'' \wedge Q(S'', L'')]]]]$ (Hyp) $L'_1.$ $LEM2$ $\vdash [[\neg[\Box \in S] \wedge 0 < exl(S)] \rightarrow \exists C''.\exists L''.[C'' \in S'' \wedge [nonunit(C'') \wedge [L'' \in C'' \wedge Q(S, L'')]]]]$ $(\forall E\ LEM2)$ $L_2.$ H_1 $\vdash 0 < exl(S)$ (J_1) $L_3.$ H_1 $\vdash \neg[\Box \in S]$ (J_2) $L_4.$ H_1 $\vdash 0 < exl(S) \wedge \neg[\Box \in S]$ $(\wedge I\ L_2,\ L_3)$ $L'_5.$ $H_1, LEM2$ $\vdash \exists C''.\exists L''.[C'' \in S \wedge [nonunit(C'') \wedge [L'' \in C'' \wedge Q(S, L'')]]]$ $(\rightarrow E\ L_4,\ L'_1)$ $L'_6.$ H_3 $\vdash \exists L''.[C \in S \wedge [nonunit(C) \wedge [L'' \in C \wedge Q(S, L'')]]]$ (Hyp) $L'_7.$ H_4 $\vdash [C \in S \wedge [nonunit(C) \wedge [L \in C \wedge Q(S, L)]]]$ (Hyp) $L'_8.$ H_4 $\vdash C \in S$ $(\wedge E*\ L'_7)$ $L'_9.$ H_4 $\vdash nonunit(C)$ $(\wedge E*\ L'_7)$ $L'_{10}.$ H_4 $\vdash L \in C$ $(\wedge E*\ L'_7)$ $L''_{10}.$ H_4 $\vdash Q(S, L)$ $(\wedge E*\ L'_7)$ $L_{11}.$ H_4 $\vdash \exists n.\Box \in Res\text{-}L(S, n)$ $(OPEN)$ $L_{12}.$ H_3 $\vdash \exists n.\Box \in Res\text{-}L(S, n)$ $(\exists E\ L_6,\ L_{11})$ $L'_{13}.$ H_2 $\vdash \exists n.\Box \in Res\text{-}L(S, n)$ $(\exists E\ L'_5,\ L_{12})$
Procedural Content	`schema – interpreter`

Figure 5: The `apply-lock-lem2` method

we omit here. In contrast to the adaptation of `apply-lem2` the predicate $Q(S, L)$ must hold before the method `apply-lock-lem2A` is applicable, that is, a supporting line with this predicate must already be in the actual planning state. This means for the method `apply-lock-lem2A` that an unmarked schematic line, like L''_{10} in `apply-lock-lem2`, must be added to the specification.

After this interactive adaptation, we can try to execute the new proof plan in order to generate a proof for lock resolution. This proof plan can be successfully executed and generates a completeness proof for lock resolution without any search at all. That is, in our framework it is possible to transfer the first proof to the second with some few user interactions only.

4 Conclusion and Future Development

The problem solving competence of a mathematician relies heavily on his/her ability to adapt problem solving knowledge to new situations where existing methods are not directly applicable. Up to now this has not received enough attention in the field of automated theorem proving. In order to mechanise parts of this ability, in earlier work we have proposed a declarative extension to the proof planning approach developed by Bundy.

This paper is aimed to provide evidence that our goal can be achieved with our declarative approach. To do this, we have presented methods, by which a proof planner can automatically generate a proof plan that can show the ground completeness of resolution. These methods form the basic repertoire that can be used to generate proofs in the meta-theory of resolution. Finding such a kind of reasoning know-how is a particularly hard task and requires a lot of genuine creativity. We did not address the problem *how* such major breakthroughs can be made automatically. In our framework the task of providing these methods has to be fulfilled by the human user; speaking in terms of knowledge-based systems, the basic knowledge in an area has to be acquired by knowledge engineers using the expertise of people working in the area. But the knowledge should be applicable in a wider range of examples than for what it was originally acquired. In the concrete example, the first proof can be generated fully automatically in the interactive setting of Ω-MKRP, once these methods are given. The five lemmata about the properties of $\subset$ can be proved by calling an automated theorem prover. In addition to automatically proving the first problem with the given methods, we were able to show that most of the methods can be reused and that the proof of ground completeness of resolution can be easily transferred with some user interactions only.

To enlarge the problem solving strength of the system, a comprising repertoire of methods is necessary. To come up with novel methods, meta-methods can be incorporated into the planning algorithm. To do this, first it must be decided when to interrupt the process of planning with methods, in order to create a new method with meta-methods. Second, for the current proof situation an adequate pair of a method and a meta-method has to be chosen from the knowledge base. We believe that there can hardly be any general answer to this problem and we have to rely on heuristics. In an interactive proof development en-

vironment like Ω-MKRP [14] users might want to make this choice by themselves while the system provides only heuristic support. Even more challenging would be an automation, of course. A trivial answer would be to apply all existing meta-methods on all existing methods and then choose heuristically the best. Such a procedure can be fairly expensive applied to a large knowledge base. The first two heuristics for choosing a method to adapt we will investigate are: first, organise methods in a hierarchy of mathematical theories and prefer methods that belong to the same theory as the current problem or whose theory is close to that of the problem in the hierarchy. Second, use general conflict solving strategies like those of OPS5, for instance, favour the methods and meta-methods with the most specific specification. Of course only *successful* methods generated in a short-term memory are integrated into the permanent knowledge base of methods.

Admittedly, the heuristic control concerning the choice of methods and meta-methods with a specific instantiation, as well as the interleaving of planning and meta-level planning, remain as open problems.

Acknowledgements

We would like to thank our colleagues, especially Xiaorong Huang and Lassaad Cheikhrouhou, for many fruitful and inspiring discussions as well as for the joint development of the declarative approach to proof planning. In addition, we would like to thank Erica Melis for many discussions on proof planning and for pointing us to Bledsoe's work about the automation of the proof of ground completeness of resolution.

A The proofs for resolution completeness

In the following we present the proofs found by the planning system which is integrated into the proof development environment Ω-MKRP. The first proof shows the completeness of binary resolution in the variable-free case.

L1.	$\mathcal{H}_1$	$\vdash [[\mathit{finite}(C_1) \wedge [\mathit{unsat}(C_1) \wedge [0 < \mathit{exl}(C_1) \wedge [\neg[\Box \in C_1] \wedge$ $\forall S'.[[\mathit{finite}(S') \wedge [\mathit{unsat}(S') \wedge \mathit{exl}(S') < \mathit{exl}(C_1)]] \rightarrow$ $\exists n'.\Box \in \mathit{Res}(S', n')]]]]] \rightarrow \exists n.\Box \in \mathit{Res}(C_1, n)]$	($\rightarrow I$ L3)
L2.	L2	$\vdash [\mathit{finite}(C_1) \wedge [\mathit{unsat}(C_1) \wedge [0 < \mathit{exl}(C_1) \wedge$ $[\neg[\Box \in C_1] \wedge \forall S'.[[\mathit{finite}(S') \wedge [\mathit{unsat}(S') \wedge$ $\mathit{exl}(S') < \mathit{exl}(C_1)]] \rightarrow \exists n'.\Box \in \mathit{Res}(S', n')]]]]]$	(Hyp)
L3.	$\mathcal{H}_2$	$\vdash \exists n.\Box \in \mathit{Res}(C_1, n)$	($\exists E$ L14,L21)
L4.	L2	$\vdash \mathit{finite}(C_1)$	($\wedge E$ L2)
L5.	L2	$\vdash [\mathit{unsat}(C_1) \wedge [0 < \mathit{exl}(C_1) \wedge [\neg[\Box \in C_1] \wedge$ $\forall S'.[[\mathit{finite}(S') \wedge [\mathit{unsat}(S') \wedge \mathit{exl}(S') < \mathit{exl}(C_1)]]$ $\rightarrow \exists n'.\Box \in \mathit{Res}(S', n')]]]]$	($\wedge E$ L2)
L6.	L2	$\vdash \mathit{unsat}(C_1)$	($\wedge E$ L5)
L7.	L2	$\vdash [0 < \mathit{exl}(C_1) \wedge [\neg[\Box \in C_1] \wedge \forall S'.[[\mathit{finite}(S') \wedge$ $[\mathit{unsat}(S') \wedge \mathit{exl}(S') < \mathit{exl}(C_1)]] \rightarrow$ $\exists n'.\Box \in \mathit{Res}(S', n')]]]$	($\wedge E$ L5)

L8.	L2	$\vdash\ 0 < \mathit{exl}(C_1)$	($\wedge E$ L7)
L9.	L2	$\vdash\ [\neg[\Box \in C_1] \wedge \forall S'.[[\mathit{finite}(S') \wedge [\mathit{unsat}(S') \wedge \mathit{exl}(S') < \mathit{exl}(C_1)]] \rightarrow \exists n'.\Box \in \mathit{Res}(S', n')]]$	($\wedge E$ L7)
L10.	L2	$\vdash\ \neg[\Box \in C_1]$	($\wedge E$ L9)
L11.	L2	$\vdash\ \forall S'.[[\mathit{finite}(S') \wedge [\mathit{unsat}(S') \wedge \mathit{exl}(S') < \mathit{exl}(C_1)]] \rightarrow \exists n'.\Box \in \mathit{Res}(S', n')]$	($\wedge E$ L9)
L12.	LEM2	$\vdash\ [[\neg[\Box \in C_1] \wedge 0 < \mathit{exl}(C_1)] \rightarrow \exists C.\exists L.[C \in C_1 \wedge [\mathit{nonunit}(C) \wedge L \in C]]]$	($\forall E$ LEM2)
L13.	L2	$\vdash\ [\neg[\Box \in C_1] \wedge 0 < \mathit{exl}(C_1)]$	($\wedge I$ L10,L8)
L14.	LEM2, L2	$\vdash\ \exists C.\exists L.[C \in S \wedge [\mathit{nonunit}(C) \wedge L \in C]]$	($\rightarrow E$ L13,L12)
L15.	$\mathcal{H}_3$	$\vdash\ \exists L.[C_2 \in C_1 \wedge [\mathit{nonunit}(C_2) \wedge L \in C_2]]$	(Hyp)
L16.	$\mathcal{H}_4$	$\vdash\ [C_2 \in C_1 \wedge [\mathit{nonunit}(C_2) \wedge C_3 \in C_2]]$	(Hyp)
L17.	$\mathcal{H}_4$	$\vdash\ C_2 \in C_1$	($\wedge E*$ L16)
L18.	$\mathcal{H}_4$	$\vdash\ \mathit{nonunit}(C_2)$	($\wedge E*$ L16)
L19.	$\mathcal{H}_4$	$\vdash\ C_3 \in C_2$	($\wedge E*$ L16)
L20.	$\mathcal{H}_4$	$\vdash\ \exists n.\Box \in \mathit{Res}(C_1, n)$	($\exists E$ L24,L33)
L21.	$\mathcal{H}_3$	$\vdash\ \exists n.\Box \in \mathit{Res}(C_1, n)$	($\exists E$ L15,L20)
L22.	LEM2A	$\vdash\ [[C_2 \in C_1 \wedge [\mathit{nonunit}(C_2) \wedge C_3 \in C_2]] \rightarrow \exists S_1.\exists S_2.\exists C_1.\exists C_2.[C_1 = C_2 \setminus_{cl} \{C_3\} \wedge [C_2 = \Box \cup_{cl} \{C_3\} \wedge [S_1 = C_1 \setminus \{C_2\} \cup \{C_1\} \wedge S_2 = C_1 \setminus \{C_2\} \cup \{C_2\}]]]]$	($\forall E*$ LEM2A)
L23.	$\mathcal{H}_4$	$\vdash\ [C_2 \in C_1 \wedge [\mathit{nonunit}(C_2) \wedge C_3 \in C_2]]$	($\wedge I*$ L17,L18,L19)
L24.	$\mathcal{H}_4$	$\vdash\ \exists S_1.\exists S_2.\exists C_1.\exists C_2.[C_1 = C_2 \setminus_{cl} \{C_3\} \wedge [C_2 = \Box \cup_{cl} \{C_3\} \wedge [S_1 = C_1 \setminus \{C_2\} \cup \{C_1\} \wedge S_2 = C_1 \setminus \{C_2\} \cup \{C_2\}]]]$	($\rightarrow E$ L23,L22)
L25.	$\mathcal{H}_5$	$\vdash\ \exists S_2.\exists C_1.\exists C_2.[C_1 = C_2 \setminus_{cl} \{C_3\} \wedge [C_2 = \Box \cup_{cl} \{C_3\} \wedge [C_6 = C_1 \setminus \{C_2\} \cup \{C_1\} \wedge S_2 = C_1 \setminus \{C_2\} \cup \{C_2\}]]]$	(Hyp)
L26.	$\mathcal{H}_6$	$\vdash\ \exists C_1.\exists C_2.[C_1 = C_2 \setminus_{cl} \{C_3\} \wedge [C_2 = \Box \cup_{cl} \{C_3\} \wedge [C_6 = C_1 \setminus \{C_2\} \cup \{C_1\} \wedge C_7 = C_1 \setminus \{C_2\} \cup \{C_2\}]]]$	(Hyp)
L27.	$\mathcal{H}_7$	$\vdash\ \exists C_2.[C_4 = C_2 \setminus_{cl} \{C_3\} \wedge [C_2 = \Box \cup_{cl} \{C_3\} \wedge [C_6 = C_1 \setminus \{C_2\} \cup \{C_4\} \wedge C_7 = C_1 \setminus \{C_2\} \cup \{C_2\}]]]$	(Hyp)
L28.	$\mathcal{H}_8$	$\vdash\ [C_4 = C_2 \setminus_{cl} \{C_3\} \wedge [C_5 = \Box \cup_{cl} \{C_3\} \wedge [C_6 = C_1 \setminus \{C_2\} \cup \{C_4\} \wedge C_7 = C_1 \setminus \{C_2\} \cup \{C_5\}]]]$	(Hyp)
L29.	$\mathcal{H}_8$	$\vdash\ C_4 = C_2 \setminus_{cl} \{C_3\}$	($\wedge E*$ L28)
L30.	$\mathcal{H}_8$	$\vdash\ C_5 = \Box \cup_{cl} \{C_3\}$	($\wedge E*$ L28)
L31.	$\mathcal{H}_8$	$\vdash\ C_6 = C_1 \setminus \{C_2\} \cup \{C_4\}$	($\wedge E*$ L28)
L32.	$\mathcal{H}_8$	$\vdash\ C_7 = C_1 \setminus \{C_2\} \cup \{C_5\}$	($\wedge E*$ L28)
L33.	$\mathcal{H}_8$	$\vdash\ \exists n.\Box \in \mathit{Res}(C_1, n)$	($\exists E$ L59,L67)
L34.	LEM0	$\vdash\ [[\mathit{finite}(C_1) \wedge [\mathit{unsat}(C_1) \wedge \mathit{subsumes}(C_1 \setminus \{C_2\} \cup \{C_2 \setminus_{cl} \{C_3\}\}, C_1)]] \rightarrow [\mathit{finite}(C_1 \setminus \{C_2\} \cup \{C_2 \setminus_{cl} \{C_3\}\}) \wedge \mathit{unsat}(C_1 \setminus \{C_2\} \cup \{C_2 \setminus_{cl} \{C_3\}\})]]$	($\forall E*$ LEM0)
L35.	$\mathcal{H}_8$	$\vdash\ \mathit{subsumes}(C_1 \setminus \{C_2\} \cup \{C_2 \setminus_{cl} \{C_3\}\}, C_1)$	(*OPEN* L19, SUBS-S1-S, SUBS-S2-S)

L36.	$\mathcal{H}_8$	$\vdash [finite(C_1) \wedge [unsat(C_1) \wedge subsumes(C_1 \setminus \{C_2\} \cup \{C_2 \setminus_{cl} \{C_3\}\}, C_1)]]$	($\wedge I*$ L4,L6,L35)
L37.	$\mathcal{H}_8$	$\vdash [finite(C_1 \setminus \{C_2\} \cup \{C_2 \setminus_{cl} \{C_3\}\}) \wedge unsat(C_1 \setminus \{C_2\} \cup \{C_2 \setminus_{cl} \{C_3\}\})]$	($\rightarrow E$ L36,L34)
L38.	$\mathcal{H}_8$	$\vdash finite(C_1 \setminus \{C_2\} \cup \{C_2 \setminus_{cl} \{C_3\}\})$	($\wedge E$ L37)
L39.	$\mathcal{H}_8$	$\vdash unsat(C_1 \setminus \{C_2\} \cup \{C_2 \setminus_{cl} \{C_3\}\})$	($\wedge E$ L37)
L40.	LEM3	$\vdash [[C_2 \in C_1 \wedge C_2 \setminus_{cl} \{C_3\} \subseteq C_2] \rightarrow exl(C_1 \setminus \{C_2\} \cup \{C_2 \setminus_{cl} \{C_3\}\}) < exl(C_1)]$	($\forall E*$ LEM3)
L41.	$\mathcal{H}_8$	$\vdash C_2 \setminus_{cl} \{C_3\} \subseteq C_2$	(*OPEN* L19,L18, $\subset$-Prop1,$\subset$-Prop3, $\subseteq$-Def,$\setminus$-Def)
L42.	$\mathcal{H}_4$	$\vdash [C_2 \in C_1 \wedge C_2 \setminus_{cl} \{C_3\} \subseteq C_2]$	($\wedge I$ L17,L41)
L43.	$\mathcal{H}_4$	$\vdash exl(C_1 \setminus \{C_2\} \cup \{C_2 \setminus_{cl} \{C_3\}\}) < exl(C_1)$	($\rightarrow E$ L42,L40)
L44.	LEM0	$\vdash [[finite(C_1) \wedge [unsat(C_1) \wedge subsumes(C_1 \setminus \{C_2\} \cup \{\square \cup_{cl} \{C_3\}\}, C_1)]] \rightarrow [finite(C_1 \setminus \{C_2\} \cup \{\square \cup_{cl} \{C_3\}\}) \wedge unsat(C_1 \setminus \{C_2\} \cup \{\square \cup_{cl} \{C_3\}\})]]$	($\forall E*$ LEM0)
L45.	$\mathcal{H}_8$	$\vdash subsumes(C_1 \setminus \{C_2\} \cup \{\square \cup_{cl} \{C_3\}\}, C_1)$	(*OPEN* L19,SUBS-S1-S, SUBS-S2-S)
L46.	$\mathcal{H}_8$	$\vdash [finite(C_1) \wedge [unsat(C_1) \wedge subsumes(C_1 \setminus \{C_2\} \cup \{\square \cup_{cl} \{C_3\}\}, C_1)]]$	($\wedge I*$ L4,L6,L45)
L47.	$\mathcal{H}_8$	$\vdash [finite(C_1 \setminus \{C_2\} \cup \{\square \cup_{cl} \{C_3\}\}) \wedge unsat(C_1 \setminus \{C_2\} \cup \{\square \cup_{cl} \{C_3\}\})]$	($\rightarrow E$ L46,L44)
L48.	$\mathcal{H}_8$	$\vdash finite(C_1 \setminus \{C_2\} \cup \{\square \cup_{cl} \{C_3\}\})$	($\wedge E$ L47)
L49.	$\mathcal{H}_8$	$\vdash unsat(C_1 \setminus \{C_2\} \cup \{\square \cup_{cl} \{C_3\}\})$	($\wedge E$ L47)
L50.	LEM3	$\vdash [[C_2 \in C_1 \wedge \square \cup_{cl} \{C_3\} \subseteq C_2] \rightarrow exl(C_1 \setminus \{C_2\} \cup \{\square \cup_{cl} \{C_3\}\}) < exl(C_1)]$	($\forall E*$ LEM3)
L51.	$\mathcal{H}_8$	$\vdash \square \cup_{cl} \{C_3\} \subseteq C_2$	(*OPEN* L19,L18, $\subset$-Prop1,$\subset$-Prop3, $\subseteq$-Def,$\setminus$-Def)
L52.	$\mathcal{H}_4$	$\vdash [C_2 \in C_1 \wedge \square \cup_{cl} \{C_3\} \subseteq C_2]$	($\wedge I$ L17,L51)
L53.	$\mathcal{H}_4$	$\vdash exl(C_1 \setminus \{C_2\} \cup \{\square \cup_{cl} \{C_3\}\}) < exl(C_1)$	($\rightarrow E$ L52,L50)
L54.	L2	$\vdash [[finite(C_1 \setminus \{C_2\} \cup \{\square \cup_{cl} \{C_3\}\}) \wedge [unsat(C_1 \setminus \{C_2\} \cup \{\square \cup_{cl} \{C_3\}\}) \wedge exl(C_1 \setminus \{C_2\} \cup \{\square \cup_{cl} \{C_3\}\}) < exl(C_1)]] \rightarrow \exists n'.\square \in Res(C_1 \setminus \{C_2\} \cup \{\square \cup_{cl} \{C_3\}\}, n')]$	($\forall E$ L11)
L55.	$\mathcal{H}_4$	$\vdash [finite(C_1 \setminus \{C_2\} \cup \{\square \cup_{cl} \{C_3\}\}) \wedge [unsat(C_1 \setminus \{C_2\} \cup \{\square \cup_{cl} \{C_3\}\}) \wedge exl(C_1 \setminus \{C_2\} \cup \{\square \cup_{cl} \{C_3\}\}) < exl(C_1)]]$	($\wedge I*$ L48,L49,L53)
L56.	$\mathcal{H}_4$	$\vdash \exists n'.\square \in Res(C_1 \setminus \{C_2\} \cup \{\square \cup_{cl} \{C_3\}\}, n')$	($\rightarrow E$ L55,L54)
L57.	L2	$\vdash [[finite(C_1 \setminus \{C_2\} \cup \{C_2 \setminus_{cl} \{C_3\}\}) \wedge [unsat(C_1 \setminus \{C_2\} \cup \{C_2 \setminus_{cl} \{C_3\}\}) \wedge exl(C_1 \setminus \{C_2\} \cup \{C_2 \setminus_{cl} \{C_3\}\}) < exl(C_1)]] \rightarrow \exists n'.\square \in Res(C_1 \setminus \{C_2\} \cup \{C_2 \setminus_{cl} \{C_3\}\}, n')]$	($\forall E$ L11)
L58.	$\mathcal{H}_4$	$\vdash [finite(C_1 \setminus \{C_2\} \cup \{C_2 \setminus_{cl} \{C_3\}\}) \wedge [unsat(C_1 \setminus \{C_2\} \cup \{C_2 \setminus_{cl} \{C_3\}\}) \wedge exl(C_1 \setminus \{C_2\} \cup \{C_2 \setminus_{cl} \{C_3\}\}) < exl(C_1)]]$	($\wedge I*$ L38,L39,L43)
L59.	$\mathcal{H}_4$	$\vdash \exists n'.\square \in Res(C_1 \setminus \{C_2\} \cup \{C_2 \setminus_{cl} \{C_3\}\}, n')$	($\rightarrow E$ L58,L57)

L60.	$\mathcal{H}_9$	$\vdash \Box \in Res(C_1 \setminus\{C_2\} \cup \{C_2 \setminus_{cl}\{C_3\}\}, C_8)$	(Hyp)
L61.	LEM1	$\vdash [\Box \in Res(C_1 \setminus\{C_2\} \cup \{C_2 \setminus_{cl}\{C_3\}\}, C_8) \rightarrow$ $[\Box \in Res(C_1 \setminus\{C_2\} \cup \{C_2 \setminus_{cl}\{C_3\} \cup_{cl} \{C_3\}\}, C_8) \vee$ $\Box \cup_{cl} \{C_3\} \in Res(C_1 \setminus\{C_2\}$ $\cup \{C_2 \setminus_{cl}\{C_3\} \cup_{cl} \{C_3\}\}, C_8)]]$	($\forall E*$ LEM1)
L62.	$\mathcal{H}_9$	$\vdash [\Box \in Res(C_1 \setminus\{C_2\} \cup \{C_2 \setminus_{cl}\{C_3\} \cup_{cl} \{C_3\}\}, C_8) \vee$ $\Box \cup_{cl} \{C_3\} \in Res(C_1 \setminus\{C_2\}$ $\cup \{C_2 \setminus_{cl}\{C_3\} \cup_{cl} \{C_3\}\}, C_8)]$	($\rightarrow E$ L60,L61)
L63.	$\mathcal{H}_{10}$	$\vdash \Box \in Res(C_1 \setminus\{C_2\} \cup \{C_2 \setminus_{cl}\{C_3\} \cup_{cl} \{C_3\}\}, C_8)$	(Case 1)
L64.	$\mathcal{H}_{10}$	$\vdash \exists n.\Box \in Res(C_1, n)$	($\exists I$ L70)
L65.	$\mathcal{H}_{11}$	$\vdash \Box \cup_{cl} \{C_3\} \in Res(C_1 \setminus\{C_2\} \cup \{C_2 \setminus_{cl}\{C_3\}$ $\cup_{cl} \{C_3\}\}, C_8)$	(Case 2)
L66.	$\mathcal{H}_{11}$	$\vdash \exists n.\Box \in Res(C_1, n)$	($\exists E$ L56,L84)
L67.	$\mathcal{H}_9$	$\vdash \exists n.\Box \in Res(C_1, n)$	($\vee E$ L62,L64,L66)
L68.	$\subset$-Prop2	$\vdash C_1 \setminus\{C_2\} \cup \{C_2 \setminus_{cl}\{C_3\} \cup_{cl} \{C_3\}\} = C_1$	($\forall E*$ $\subset$-Prop2)
L69.	$\mathcal{H}_{10}$	$\vdash \Box \in Res(C_1, C_8)$	(=Subst L68,L63)
L70.	$\mathcal{H}_{10}$	$\vdash \Box \in Res(C_1, C_8)$	(Same L69)
L71.	LEM4	$\vdash C_1 \subseteq Res(C_1, C_8)$	($\forall E*$ LEM4)
L72.	$\mathcal{H}_{11}$	$\vdash \forall V_3.\forall V_4.V_3 \setminus\{V_4\} \subseteq V_3$	($OPEN$ $\subseteq$-Def,\-Def)
L73.	$\mathcal{H}_{11}$	$\vdash C_1 \setminus\{C_2\} \subseteq C_1$	($\forall E*$ L72)
L74.	$\mathcal{H}_{11}$	$\vdash [C_1 \setminus\{C_2\} \subseteq C_1 \wedge C_1 \subseteq Res(C_1, C_8)]$	($\wedge I$ L73,L71)
L75.	$\subseteq$-Trans	$\vdash [[C_1 \setminus\{C_2\} \subseteq C_1 \wedge C_1 \subseteq Res(C_1, C_8)] \rightarrow$ $C_1 \setminus\{C_2\} \subseteq Res(C_1, C_8)]$	($\forall E*$ $\subseteq$-Trans)
L76.	$\mathcal{H}_{11}$	$\vdash C_1 \setminus\{C_2\} \subseteq Res(C_1, C_8)$	($\rightarrow E$ L74,L75)
L77.	LEM-13	$\vdash [[\Box \cup_{cl} \{C_3\} \in Res(C_1, C_8) \wedge$ $C_1 \setminus\{C_2\} \subseteq Res(C_1, C_8)] \rightarrow$ $C_1 \setminus\{C_2\} \cup \{\Box \cup_{cl} \{C_3\}\} \subseteq Res(C_1, C_8)]$	($\forall E*$ LEM-13)
L78.	$\mathcal{H}_{11}$	$\vdash \Box \cup_{cl} \{C_3\} \in Res(C_1, C_8)$	(=Subst L68,L65)
L79.	$\mathcal{H}_{11}$	$\vdash [\Box \cup_{cl} \{C_3\} \in Res(C_1, C_8) \wedge C_1 \setminus\{C_2\} \subseteq Res(C_1, C_8)]$	($\wedge I$ L78,L76)
L80.	$\mathcal{H}_{11}$	$\vdash C_1 \setminus\{C_2\} \cup \{\Box \cup_{cl} \{C_3\}\} \subseteq Res(C_1, C_8)$	($\rightarrow E$ L79,L77)
L81.	$\mathcal{H}_8$	$\vdash C_7 = C_1 \setminus\{C_2\} \cup \{\Box \cup_{cl} \{C_3\}\}$	(=Subst L30,L32)
L82.	$\mathcal{H}_{11}$	$\vdash C_7 \subseteq Res(C_1, C_8)$	(=Subst L81,L80)
L83.	$\mathcal{H}_{12}$	$\vdash \Box \in Res(C_1, C_9)$	(Hyp)
L84.	$\mathcal{H}_{12}$	$\vdash \exists n.\Box \in Res(C_1, n)$	($\exists I$ L92)
L85.	LEM5	$\vdash [C_7 \subseteq Res(C_1, C_8) \rightarrow$ $Res(C_7, C_9) \subseteq Res(Res(C_1, C_8), C_9)]$	($\forall E*$ LEM5)
L86.	$\mathcal{H}_{11}$	$\vdash Res(C_7, C_9) \subseteq Res(Res(C_1, C_8), C_9)$	($\rightarrow E$ L82,L85)
L87.	$\mathcal{H}_{12}$	$\vdash \Box \in Res(C_7, C_9)$	(=Subst L81,L83)
L88.	LEM6	$\vdash Res(Res(C_1, C_8), C_9) = Res(C_1, C_8 + C_9)$	($\forall E*$ LEM6)
L89.	$\mathcal{H}_{11}$	$\vdash Res(C_7, C_9) \subseteq Res(C_1, C_8 + C_9)$	(=Subst L88,L86)
L90.	LEM-14	$\vdash [[\Box \in Res(C_7, C_9) \wedge Res(C_7, C_9) \subseteq Res(C_1, C_8 + C_9)]$ $\rightarrow \Box \in Res(C_1, C_8 + C_9)]$	($\forall E*$ LEM-14)
L91.	$\mathcal{H}_{12}$	$\vdash \Box \in Res(C_1, C_8 + C_9)$	($OPEN$ L87,L89,L90)

L92.	$\mathcal{H}_{12}$	$\vdash \Box \in Res(C_1, C_8 + C_9)$	(Same L91)
THM.	$\mathcal{H}_1$	$\vdash \forall S.[[finite(S) \wedge [unsat(S) \wedge [0 < exl(S) \wedge [\neg[\Box \in S] \wedge$ $\forall S'.[[finite(S') \wedge [unsat(S') \wedge exl(S') < exl(S)]] \rightarrow$ $\exists n'.\Box \in Res(S', n')]]]]] \rightarrow \exists n.\Box \in Res(S, n)]$	($\forall I*$ L1)

The hypotheses of the proof lines are abbreviated as follows: $\mathcal{H}_1$ are all axiom lines of the proof; $\mathcal{H}_2 = \mathcal{H}_1$, L2; $\mathcal{H}_3 = \mathcal{H}_2$, L15; $\mathcal{H}_4 = \mathcal{H}_3$, L16; $\mathcal{H}_5 = \mathcal{H}_4$, L25; $\mathcal{H}_6 = \mathcal{H}_5$, L26; $\mathcal{H}_7 = \mathcal{H}_6$, L27; $\mathcal{H}_8 = \mathcal{H}_7$, L28; $\mathcal{H}_9 = \mathcal{H}_8$, L60; $\mathcal{H}_{10} = \mathcal{H}_9$, L63; $\mathcal{H}_{11} = \mathcal{H}_9$, L65; $\mathcal{H}_{12} = \mathcal{H}_{11}$, L83;

Note that in the proof above there are still unproved open lines: L35, L41, L45, L51, L72, and L91. But these depend only on a small set of hypotheses and can either be proved by calling an automated theorem prover (e.g. OTTER) interactively, or by a hierarchical planning process which takes each open line and its corresponding hypotheses as independent planning subproblems.

The proof shown next is the proof generated by applying the altered proof plan to the initial state for locking resolution.

L1.	$\mathcal{H}_1$	$\vdash [[finite(C_{10}) \wedge [unsat\text{-}L(C_{10}) \wedge [0 < exl(C_{10}) \wedge$ $[\neg[\Box \in C_{10}] \wedge \forall S'.[[finite(S') \wedge [unsat\text{-}L(S') \wedge$ $exl(S') < exl(C_{10})]] \rightarrow \exists n'.\Box \in Res\text{-}L(S', n')]]]]] \rightarrow$ $\exists n.\Box \in Res\text{-}L(C_{10}, n)]$	($\rightarrow I$ L3)
L2.	L2	$\vdash [finite(C_{10}) \wedge [unsat\text{-}L(C_{10}) \wedge [0 < exl(C_{10}) \wedge$ $[\neg[\Box \in C_{10}] \wedge \forall S'.[[finite(S') \wedge [unsat\text{-}L(S') \wedge$ $exl(S') < exl(C_{10})]] \rightarrow \exists n'.\Box \in Res\text{-}L(S', n')]]]]]$	(Hyp)
L3.	$\mathcal{H}_2$	$\vdash \exists n.\Box \in Res\text{-}L(C_{10}, n)$	($\exists E$ L14,L22)
L4.	L2	$\vdash finite(C_{10})$	($\wedge E$ L2)
L5.	L2	$\vdash [unsat\text{-}L(C_{10}) \wedge [0 < exl(C_{10}) \wedge [\neg[\Box \in C_{10}] \wedge$ $\forall S'.[[finite(S') \wedge [unsat\text{-}L(S') \wedge exl(S') < exl(C_{10})]]$ $\rightarrow \exists n'.\Box \in Res\text{-}L(S', n')]]]]$	($\wedge E$ L2)
L6.	L2	$\vdash unsat\text{-}L(C_{10})$	($\wedge E$ L5)
L7.	L2	$\vdash [0 < exl(C_{10}) \wedge [\neg[\Box \in C_{10}] \wedge \forall S'.[[finite(S') \wedge$ $[unsat\text{-}L(S') \wedge exl(S') < exl(C_{10})]] \rightarrow$ $\exists n'.\Box \in Res\text{-}L(S', n')]]]$	($\wedge E$ L5)
L8.	L2	$\vdash 0 < exl(C_{10})$	($\wedge E$ L7)
L9.	L2	$\vdash [\neg[\Box \in C_{10}] \wedge \forall S'.[[finite(S') \wedge [unsat\text{-}L(S') \wedge$ $exl(S') < exl(C_{10})]] \rightarrow \exists n'.\Box \in Res\text{-}L(S', n')]]$	($\wedge E$ L7)
L10.	L2	$\vdash \neg[\Box \in C_{10}]$	($\wedge E$ L9)
L11.	L2	$\vdash \forall S'.[[finite(S') \wedge [unsat\text{-}L(S') \wedge exl(S') < exl(C_{10})]] \rightarrow$ $\exists n'.\Box \in Res\text{-}L(S', n')]$	($\wedge E$ L9)
L12.	LEM2	$\vdash [[\neg[\Box \in C_{10}] \wedge 0 < exl(C_{10})] \rightarrow \exists C.\exists L.$ $[C \in C_{10} \wedge [nonunit(C) \wedge [L \in C \wedge Q(C_{10}, L)]]]]$	($\forall E$ LEM2)
L13.	L2	$\vdash [\neg[\Box \in C_{10}] \wedge 0 < exl(C_{10})]$	($\wedge I$ L10,L8)
L14.	LEM2, L2	$\vdash \exists C.\exists L.[C \in S \wedge [nonunit(C) \wedge [L \in C \wedge Q(S, L)]]]$	($\rightarrow E$ L13,L12)
L15.	$\mathcal{H}_3$	$\vdash \exists L.[C_{11} \in C_{10} \wedge [nonunit(C_{11}) \wedge [L \in C_{11} \wedge Q(C_{10}, L)]]]$	(Hyp)
L16.	$\mathcal{H}_4$	$\vdash [C_{11} \in C_{10} \wedge [nonunit(C_{11}) \wedge [C_{12} \in C_{11} \wedge Q(C_{10}, C_{12})]]]$	(Hyp)
L17.	$\mathcal{H}_4$	$\vdash C_{11} \in C_{10}$	($\wedge E*$ L16)

L18.	$\mathcal{H}_4$	$\vdash$ $nonunit(C_{11})$	($\wedge E*$ L16)
L19.	$\mathcal{H}_4$	$\vdash$ $C_{12} \in C_{11}$	($\wedge E*$ L16)
L20.	$\mathcal{H}_4$	$\vdash$ $Q(C_{10}, C_{12})$	($\wedge E*$ L16)
L21.	$\mathcal{H}_4$	$\vdash$ $\exists n.\Box \in Res\text{-}L(C_{10}, n)$	($\exists E$ L25,L34)
L22.	$\mathcal{H}_3$	$\vdash$ $\exists n.\Box \in Res\text{-}L(C_{10}, n)$	($\exists E$ L15,L21)
L23.	LEM2A	$\vdash$ $[[C_{11} \in C_{10} \wedge [nonunit(C_{11}) \wedge [C_{12} \in C_{11} \wedge$ $Q(C_{10}, C_{12})]]] \rightarrow \exists S_1.\exists S_2.\exists C_1.$ $\exists C_2.[C_1 = C_{11} \setminus_{cl} \{C_{12}\} \wedge [C_2 = \Box \cup_{cl} \{C_{12}\} \wedge$ $[S_1 = C_{10} \setminus \{C_{11}\} \cup \{C_1\} \wedge$ $S_2 = C_{10} \setminus \{C_{11}\} \cup \{C_2\}]]]]$	($\forall E*$ LEM2A)
L24.	$\mathcal{H}_4$	$\vdash$ $[C_{11} \in C_{10} \wedge [nonunit(C_{11}) \wedge [C_{12} \in C_{11} \wedge Q(C_{10}, C_{12})]]]$	($\wedge I*$ L17,L18,L19,L20)
L25.	$\mathcal{H}_4$	$\vdash$ $\exists S_1.\exists S_2.\exists C_1.\exists C_2.[C_1 = C_{11} \setminus_{cl} \{C_{12}\} \wedge$ $[C_2 = \Box \cup_{cl} \{C_{12}\} \wedge [S_1 = C_{10} \setminus \{C_{11}\} \cup \{C_1\} \wedge$ $S_2 = C_{10} \setminus \{C_{11}\} \cup \{C_2\}]]]$	($\rightarrow E$ L24,L23)
L26.	$\mathcal{H}_5$	$\vdash$ $\exists S_2.\exists C_1.\exists C_2.[C_1 = C_{11} \setminus_{cl} \{C_{12}\} \wedge [C_2 = \Box \cup_{cl} \{C_{12}\} \wedge$ $[C_{15} = C_{10} \setminus \{C_{11}\} \cup \{C_1\} \wedge$ $S_2 = C_{10} \setminus \{C_{11}\} \cup \{C_2\}]]]$	(Hyp)
L27.	$\mathcal{H}_6$	$\vdash$ $\exists C_1.\exists C_2.[C_1 = C_{11} \setminus_{cl} \{C_{12}\} \wedge [C_2 = \Box \cup_{cl} \{C_{12}\} \wedge$ $[C_{15} = C_{10} \setminus \{C_{11}\} \cup \{C_1\} \wedge$ $C_{16} = C_{10} \setminus \{C_{11}\} \cup \{C_2\}]]]$	(Hyp)
L28.	$\mathcal{H}_7$	$\vdash$ $\exists C_2.[C_{13} = C_{11} \setminus_{cl} \{C_{12}\} \wedge [C_2 = \Box \cup_{cl} \{C_{12}\} \wedge$ $[C_{15} = C_{10} \setminus \{C_{11}\} \cup \{C_{13}\} \wedge$ $C_{16} = C_{10} \setminus \{C_{11}\} \cup \{C_2\}]]]$	(Hyp)
L29.	$\mathcal{H}_8$	$\vdash$ $[C_{13} = C_{11} \setminus_{cl} \{C_{12}\} \wedge [C_{14} = \Box \cup_{cl} \{C_{12}\} \wedge$ $[C_{15} = C_{10} \setminus \{C_{11}\} \cup \{C_{13}\} \wedge$ $C_{16} = C_{10} \setminus \{C_{11}\} \cup \{C_{14}\}]]]$	(Hyp)
L30.	$\mathcal{H}_8$	$\vdash$ $C_{13} = C_{11} \setminus_{cl} \{C_{12}\}$	($\wedge E*$ L29)
L31.	$\mathcal{H}_8$	$\vdash$ $C_{14} = \Box \cup_{cl} \{C_{12}\}$	($\wedge E*$ L29)
L32.	$\mathcal{H}_8$	$\vdash$ $C_{15} = C_{10} \setminus \{C_{11}\} \cup \{C_{13}\}$	($\wedge E*$ L29)
L33.	$\mathcal{H}_8$	$\vdash$ $C_{16} = C_{10} \setminus \{C_{11}\} \cup \{C_{14}\}$	($\wedge E*$ L29)
L34.	$\mathcal{H}_8$	$\vdash$ $\exists n.\Box \in Res\text{-}L(C_{10}, n)$	($\exists E$ L60,L68)
L35.	LEM0	$\vdash$ $[[finite(C_{10}) \wedge [unsat\text{-}L(C_{10}) \wedge$ $subsumes(C_{10} \setminus \{C_{11}\} \cup \{C_{11} \setminus_{cl} \{C_{12}\}\}, C_{10})]] \rightarrow$ $[finite(C_{10} \setminus \{C_{11}\} \cup \{C_{11} \setminus_{cl} \{C_{12}\}\}) \wedge$ $unsat\text{-}L(C_{10} \setminus \{C_{11}\} \cup \{C_{11} \setminus_{cl} \{C_{12}\}\})]]$	($\forall E*$ LEM0)
L36.	$\mathcal{H}_8$	$\vdash$ $subsumes(C_{10} \setminus \{C_{11}\} \cup \{C_{11} \setminus_{cl} \{C_{12}\}\}, C_{10})$	(*OPEN* L19,SUBSUMES-S1-S,SUBSUMES-S2-S)
L37.	$\mathcal{H}_8$	$\vdash$ $[finite(C_{10}) \wedge [unsat\text{-}L(C_{10}) \wedge$ $subsumes(C_{10} \setminus \{C_{11}\} \cup \{C_{11} \setminus_{cl} \{C_{12}\}\}, C_{10})]]$	($\wedge I*$ L4,L6,L36)
L38.	$\mathcal{H}_8$	$\vdash$ $[finite(C_{10} \setminus \{C_{11}\} \cup \{C_{11} \setminus_{cl} \{C_{12}\}\}) \wedge$ $unsat\text{-}L(C_{10} \setminus \{C_{11}\} \cup \{C_{11} \setminus_{cl} \{C_{12}\}\})]$	($\rightarrow E$ L37,L35)
L39.	$\mathcal{H}_8$	$\vdash$ $finite(C_{10} \setminus \{C_{11}\} \cup \{C_{11} \setminus_{cl} \{C_{12}\}\})$	($\wedge E$ L38)
L40.	$\mathcal{H}_8$	$\vdash$ $unsat\text{-}L(C_{10} \setminus \{C_{11}\} \cup \{C_{11} \setminus_{cl} \{C_{12}\}\})$	($\wedge E$ L38)
L41.	LEM3	$\vdash$ $[[C_{11} \in C_{10} \wedge C_{11} \setminus_{cl} \{C_{12}\} \subseteq C_{11}] \rightarrow$ $exl(C_{10} \setminus \{C_{11}\} \cup \{C_{11} \setminus_{cl} \{C_{12}\}\}) < exl(C_{10})]$	($\forall E*$ LEM3)

L42.	$\mathcal{H}_8$	$\vdash C_{11} \setminus_{cl} \{C_{12}\} \subseteq C_{11}$	(*OPEN* L19, L18, SUBSET-PROP1, SUBSET-PROP3, SUBSETEQ-DEF, SETDIFF-DEF)
L43.	$\mathcal{H}_4$	$\vdash [C_{11} \in C_{10} \wedge C_{11} \setminus_{cl} \{C_{12}\} \subseteq C_{11}]$	($\wedge I$ L17, L42)
L44.	$\mathcal{H}_4$	$\vdash \mathit{exl}(C_{10} \setminus \{C_{11}\} \cup \{C_{11} \setminus_{cl} \{C_{12}\}\}) < \mathit{exl}(C_{10})$	($\rightarrow E$ L43,L41)
L45.	LEM0	$\vdash [[\mathit{finite}(C_{10}) \wedge [\mathit{unsat}\text{-}L(C_{10}) \wedge$ $\mathit{subsumes}(C_{10} \setminus \{C_{11}\} \cup \{\square \cup_{cl} \{C_{12}\}\}, C_{10})]] \rightarrow$ $[\mathit{finite}(C_{10} \setminus \{C_{11}\} \cup \{\square \cup_{cl} \{C_{12}\}\}) \wedge$ $\mathit{unsat}\text{-}L(C_{10} \setminus \{C_{11}\} \cup \{\square \cup_{cl} \{C_{12}\}\})]]$	($\forall E*$ LEM0)
L46.	$\mathcal{H}_8$	$\vdash \mathit{subsumes}(C_{10} \setminus \{C_{11}\} \cup \{\square \cup_{cl} \{C_{12}\}\}, C_{10})$	(*OPEN* L19,SUBSUMES-S1-S,SUBSUMES-S2-S)
L47.	$\mathcal{H}_8$	$\vdash [\mathit{finite}(C_{10}) \wedge [\mathit{unsat}\text{-}L(C_{10}) \wedge$ $\mathit{subsumes}(C_{10} \setminus \{C_{11}\} \cup \{\square \cup_{cl} \{C_{12}\}\}, C_{10})]]$	($\wedge I*$ L4,L6,L46)
L48.	$\mathcal{H}_8$	$\vdash [\mathit{finite}(C_{10} \setminus \{C_{11}\} \cup \{\square \cup_{cl} \{C_{12}\}\}) \wedge$ $\mathit{unsat}\text{-}L(C_{10} \setminus \{C_{11}\} \cup \{\square \cup_{cl} \{C_{12}\}\})]$	($\rightarrow E$ L47,L45)
L49.	$\mathcal{H}_8$	$\vdash \mathit{finite}(C_{10} \setminus \{C_{11}\} \cup \{\square \cup_{cl} \{C_{12}\}\})$	($\wedge E$ L48)
L50.	$\mathcal{H}_8$	$\vdash \mathit{unsat}\text{-}L(C_{10} \setminus \{C_{11}\} \cup \{\square \cup_{cl} \{C_{12}\}\})$	($\wedge E$ L48)
L51.	LEM3	$\vdash [[C_{11} \in C_{10} \wedge \square \cup_{cl} \{C_{12}\} \subseteq C_{11}] \rightarrow$ $\mathit{exl}(C_{10} \setminus \{C_{11}\} \cup \{\square \cup_{cl} \{C_{12}\}\}) < \mathit{exl}(C_{10})]$	($\forall E*$ LEM3)
L52.	$\mathcal{H}_8$	$\vdash \square \cup_{cl} \{C_{12}\} \subseteq C_{11}$	(*OPEN* L19,L18,SUBSET-PROP1,SUBSET-PROP3,SUBSETEQ-DEF,SETDIFF-DEF)
L53.	$\mathcal{H}_4$	$\vdash [C_{11} \in C_{10} \wedge \square \cup_{cl} \{C_{12}\} \subseteq C_{11}]$	($\wedge I$ L17,L52)
L54.	$\mathcal{H}_4$	$\vdash \mathit{exl}(C_{10} \setminus \{C_{11}\} \cup \{\square \cup_{cl} \{C_{12}\}\}) < \mathit{exl}(C_{10})$	($\rightarrow E$ L53,L51)
L55.	L2	$\vdash [[\mathit{finite}(C_{10} \setminus \{C_{11}\} \cup \{\square \cup_{cl} \{C_{12}\}\}) \wedge$ $[\mathit{unsat}\text{-}L(C_{10} \setminus \{C_{11}\} \cup \{\square \cup_{cl} \{C_{12}\}\}) \wedge$ $\mathit{exl}(C_{10} \setminus \{C_{11}\} \cup \{\square \cup_{cl} \{C_{12}\}\}) < \mathit{exl}(C_{10})]] \rightarrow$ $\exists n'.\square \in \mathit{Res}\text{-}L(C_{10} \setminus \{C_{11}\} \cup \{\square \cup_{cl} \{C_{12}\}\}, n')]$	($\forall E$ L11)
L56.	$\mathcal{H}_4$	$\vdash [\mathit{finite}(C_{10} \setminus \{C_{11}\} \cup \{\square \cup_{cl} \{C_{12}\}\}) \wedge$ $[\mathit{unsat}\text{-}L(C_{10} \setminus \{C_{11}\} \cup \{\square \cup_{cl} \{C_{12}\}\}) \wedge$ $\mathit{exl}(C_{10} \setminus \{C_{11}\} \cup \{\square \cup_{cl} \{C_{12}\}\}) < \mathit{exl}(C_{10})]]$	($\wedge I*$ L49,L50,L54)
L57.	$\mathcal{H}_4$	$\vdash \exists n'.\square \in \mathit{Res}\text{-}L(C_{10} \setminus \{C_{11}\} \cup \{\square \cup_{cl} \{C_{12}\}\}, n')$	($\rightarrow E$ L56,L55)
L58.	L2	$\vdash [[\mathit{finite}(C_{10} \setminus \{C_{11}\} \cup \{C_{11} \setminus_{cl} \{C_{12}\}\}) \wedge$ $[\mathit{unsat}\text{-}L(C_{10} \setminus \{C_{11}\} \cup \{C_{11} \setminus_{cl} \{C_{12}\}\}) \wedge$ $\mathit{exl}(C_{10} \setminus \{C_{11}\} \cup \{C_{11} \setminus_{cl} \{C_{12}\}\}) < \mathit{exl}(C_{10})]] \rightarrow$ $\exists n'.\square \in \mathit{Res}\text{-}L(C_{10} \setminus \{C_{11}\} \cup \{C_{11} \setminus_{cl} \{C_{12}\}\}, n')]$	($\forall E$ L11)
L59.	$\mathcal{H}_4$	$\vdash [\mathit{finite}(C_{10} \setminus \{C_{11}\} \cup \{C_{11} \setminus_{cl} \{C_{12}\}\}) \wedge$ $[\mathit{unsat}\text{-}L(C_{10} \setminus \{C_{11}\} \cup \{C_{11} \setminus_{cl} \{C_{12}\}\}) \wedge$ $\mathit{exl}(C_{10} \setminus \{C_{11}\} \cup \{C_{11} \setminus_{cl} \{C_{12}\}\}) < \mathit{exl}(C_{10})]]$	($\wedge I*$ L39,L40,L44)
L60.	$\mathcal{H}_4$	$\vdash \exists n'.\square \in \mathit{Res}\text{-}L(C_{10} \setminus \{C_{11}\} \cup \{C_{11} \setminus_{cl} \{C_{12}\}\}, n')$	($\rightarrow E$ L59,L58)
L61.	$\mathcal{H}_9$	$\vdash [\square \in \mathit{Res}\text{-}L(C_{10} \setminus \{C_{11}\} \cup \{C_{11} \setminus_{cl} \{C_{12}\}\}, C_{17}) \wedge$ $Q(C_{10}, C_{12})]$	(Hyp)

L62.	LEM1	$\vdash [[\Box \in \mathit{Res}\text{-}L(C_{10} \setminus \{C_{11}\} \cup \{C_{11} \setminus_{cl} \{C_{12}\}\}, C_{17}) \wedge Q(C_{10}, C_{12})] \rightarrow [\Box \in \mathit{Res}\text{-}L(C_{10} \setminus \{C_{11}\} \cup \{C_{11} \setminus_{cl} \{C_{12}\} \cup_{cl} \{C_{12}\}\}, C_{17}) \vee \Box \cup_{cl} \{C_{12}\} \in \mathit{Res}\text{-}L(C_{10} \setminus \{C_{11}\} \cup \{C_{11} \setminus_{cl} \{C_{12}\} \cup_{cl} \{C_{12}\}\}, C_{17})]]$	($\forall E*$ LEM1)
L63.	$\mathcal{H}_9$	$\vdash [\Box \in \mathit{Res}\text{-}L(C_{10} \setminus \{C_{11}\} \cup \{C_{11} \setminus_{cl} \{C_{12}\} \cup_{cl} \{C_{12}\}\}, C_{17}) \vee \Box \cup_{cl} \{C_{12}\} \in \mathit{Res}\text{-}L(C_{10} \setminus \{C_{11}\} \cup \{C_{11} \setminus_{cl} \{C_{12}\} \cup_{cl} \{C_{12}\}\}, C_{17})]$	($\rightarrow E$ L61,L62)
L64.	$\mathcal{H}_{10}$	$\vdash \Box \in \mathit{Res}\text{-}L(C_{10} \setminus \{C_{11}\} \cup \{C_{11} \setminus_{cl} \{C_{12}\} \cup_{cl} \{C_{12}\}\}, C_{17})$	(Case 1)
L65.	$\mathcal{H}_{10}$	$\vdash \exists n.\Box \in \mathit{Res}\text{-}L(C_{10}, n)$	($\exists I$ L71)
L66.	$\mathcal{H}_{11}$	$\vdash \Box \cup_{cl} \{C_{12}\} \in \mathit{Res}\text{-}L(C_{10} \setminus \{C_{11}\} \cup \{C_{11} \setminus_{cl} \{C_{12}\} \cup_{cl} \{C_{12}\}\}, C_{17})$	(Case 2)
L67.	$\mathcal{H}_{11}$	$\vdash \exists n.\Box \in \mathit{Res}\text{-}L(C_{10}, n)$	($\exists E$ L57,L85)
L68.	$\mathcal{H}_9$	$\vdash \exists n.\Box \in \mathit{Res}\text{-}L(C_{10}, n)$	($\vee E$ L63,L65,L67)
L69.	SUBSET-PROP2	$\vdash C_{10} \setminus \{C_{11}\} \cup \{C_{11} \setminus_{cl} \{C_{12}\} \cup_{cl} \{C_{12}\}\} = C_{10}$	($\forall E*$ SUBSET-PROP2)
L70.	$\mathcal{H}_{10}$	$\vdash \Box \in \mathit{Res}\text{-}L(C_{10}, C_{17})$	(=Subst L69,L64)
L71.	$\mathcal{H}_{10}$	$\vdash \Box \in \mathit{Res}\text{-}L(C_{10}, C_{17})$	(Same L70)
L72.	LEM4	$\vdash C_{10} \subseteq \mathit{Res}\text{-}L(C_{10}, C_{17})$	($\forall E*$ LEM4)
L73.	$\mathcal{H}_{11}$	$\vdash \forall V_{15}.\forall V_{16}.V_{15} \setminus \{V_{16}\} \subseteq V_{15}$	(*OPEN* SUBSETEQ-DEF,SETDIFF-DEF)
L74.	$\mathcal{H}_{11}$	$\vdash C_{10} \setminus \{C_{11}\} \subseteq C_{10}$	($\forall E*$ L73)
L75.	$\mathcal{H}_{11}$	$\vdash [C_{10} \setminus \{C_{11}\} \subseteq C_{10} \wedge C_{10} \subseteq \mathit{Res}\text{-}L(C_{10}, C_{17})]$	($\wedge I$ L74,L72)
L76.	SUBSETEQ-TRANS	$\vdash [[C_{10} \setminus \{C_{11}\} \subseteq C_{10} \wedge C_{10} \subseteq \mathit{Res}\text{-}L(C_{10}, C_{17})] \rightarrow C_{10} \setminus \{C_{11}\} \subseteq \mathit{Res}\text{-}L(C_{10}, C_{17})]$	($\forall E*$ SUBSETEQ-TRANS)
L77.	$\mathcal{H}_{11}$	$\vdash C_{10} \setminus \{C_{11}\} \subseteq \mathit{Res}\text{-}L(C_{10}, C_{17})$	($\rightarrow E$ L75,L76)
L78.	LEM-13	$\vdash [[\Box \cup_{cl} \{C_{12}\} \in \mathit{Res}\text{-}L(C_{10}, C_{17}) \wedge C_{10} \setminus \{C_{11}\} \subseteq \mathit{Res}\text{-}L(C_{10}, C_{17})] \rightarrow C_{10} \setminus \{C_{11}\} \cup \{\Box \cup_{cl} \{C_{12}\}\} \subseteq \mathit{Res}\text{-}L(C_{10}, C_{17})]$	($\forall E*$ LEM-13)
L79.	$\mathcal{H}_{11}$	$\vdash \Box \cup_{cl} \{C_{12}\} \in \mathit{Res}\text{-}L(C_{10}, C_{17})$	(=Subst L69,L66)
L80.	$\mathcal{H}_{11}$	$\vdash [\Box \cup_{cl} \{C_{12}\} \in \mathit{Res}\text{-}L(C_{10}, C_{17}) \wedge C_{10} \setminus \{C_{11}\} \subseteq \mathit{Res}\text{-}L(C_{10}, C_{17})]$	($\wedge I$ L79,L77)
L81.	$\mathcal{H}_{11}$	$\vdash C_{10} \setminus \{C_{11}\} \cup \{\Box \cup_{cl} \{C_{12}\}\} \subseteq \mathit{Res}\text{-}L(C_{10}, C_{17})$	($\rightarrow E$ L80,L78)
L82.	$\mathcal{H}_8$	$\vdash C_{16} = C_{10} \setminus \{C_{11}\} \cup \{\Box \cup_{cl} \{C_{12}\}\}$	(=Subst L31,L33)
L83.	$\mathcal{H}_{11}$	$\vdash C_{16} \subseteq \mathit{Res}\text{-}L(C_{10}, C_{17})$	(=Subst L82,L81)
L84.	$\mathcal{H}_{12}$	$\vdash \Box \in \mathit{Res}\text{-}L(C_{10}, C_{18})$	(Hyp)
L85.	$\mathcal{H}_{12}$	$\vdash \exists n.\Box \in \mathit{Res}\text{-}L(C_{10}, n)$	($\exists I$ L93)
L86.	LEM5	$\vdash [C_{16} \subseteq \mathit{Res}\text{-}L(C_{10}, C_{17}) \rightarrow \mathit{Res}\text{-}L(C_{16}, C_{18}) \subseteq \mathit{Res}\text{-}L(\mathit{Res}\text{-}L(C_{10}, C_{17}), C_{18})]$	($\forall E*$ LEM5)
L87.	$\mathcal{H}_{11}$	$\vdash \mathit{Res}\text{-}L(C_{16}, C_{18}) \subseteq \mathit{Res}\text{-}L(\mathit{Res}\text{-}L(C_{10}, C_{17}), C_{18})$	($\rightarrow E$ L83,L86)
L88.	$\mathcal{H}_{12}$	$\vdash \Box \in \mathit{Res}\text{-}L(C_{16}, C_{18})$	(=Subst L82,L84)
L89.	LEM6	$\vdash \mathit{Res}\text{-}L(\mathit{Res}\text{-}L(C_{10}, C_{17}), C_{18}) = \mathit{Res}\text{-}L(C_{10}, C_{17} + C_{18})$	($\forall E*$ LEM6)
L90.	$\mathcal{H}_{11}$	$\vdash \mathit{Res}\text{-}L(C_{16}, C_{18}) \subseteq \mathit{Res}\text{-}L(C_{10}, C_{17} + C_{18})$	(=Subst L89,L87)

L91.	LEM-14	$\vdash [[\Box \in \mathit{Res\text{-}L}(C_{16}, C_{18}) \wedge$ $\mathit{Res\text{-}L}(C_{16}, C_{18}) \subseteq \mathit{Res\text{-}L}(C_{10}, C_{17} + C_{18})] \rightarrow$ $\Box \in \mathit{Res\text{-}L}(C_{10}, C_{17} + C_{18})]$	($\forall E*$ LEM-14)
L92.	$\mathcal{H}_{12}$	$\vdash \Box \in \mathit{Res\text{-}L}(C_{10}, C_{17} + C_{18})$	(*OPEN* L88,L90,L91)
L93.	$\mathcal{H}_{12}$	$\vdash \Box \in \mathit{Res\text{-}L}(C_{10}, C_{17} + C_{18})$	(Same L92)
THM.	$\mathcal{H}_1$	$\vdash \forall S.[[\mathit{finite}(S) \wedge [\mathit{unsat\text{-}L}(S) \wedge [0 < \mathit{exl}(S) \wedge [\neg[\Box \in S] \wedge$ $\forall S'.[[\mathit{finite}(S') \wedge [\mathit{unsat\text{-}L}(S') \wedge \mathit{exl}(S') < \mathit{exl}(S)]]$ $\rightarrow \exists n'.\Box \in \mathit{Res\text{-}L}(S', n')]]]]] \rightarrow \exists n.\Box \in \mathit{Res\text{-}L}(S, n)]$	($\forall I*$ L1)

In this proof the hypotheses are: $\mathcal{H}_1$ is the set of all axiom lines (as in the example of the binary resolution proof); $\mathcal{H}_2 = \mathcal{H}_1$, L2; $\mathcal{H}_3 = \mathcal{H}_2$, L15; $\mathcal{H}_4 = \mathcal{H}_3$, L16; $\mathcal{H}_5 = \mathcal{H}_4$, L26; $\mathcal{H}_6 = \mathcal{H}_5$, L27; $\mathcal{H}_7 = \mathcal{H}_6$, L28; $\mathcal{H}_8 = \mathcal{H}_7$, L29; $\mathcal{H}_9 = \mathcal{H}_8$, L61; $\mathcal{H}_{10} = \mathcal{H}_9$, L64; $\mathcal{H}_{11} = \mathcal{H}_9$, L66; $\mathcal{H}_{12} = \mathcal{H}_{11}$, L84;

As in the proof of binary resolution completeness there are also open lines here which must be shown: L36, L42, L46, L52, L73, and L92. But they can be proved interactively or by further planning processes, too.

References

[1] R. Anderson and W. W. Bledsoe. A Linear Proof Format for Resolution with Merging and a New Technique for Establishing Completeness. *Journ. of ACM*, 17(3):525–534, 1970.

[2] P. B. Andrews. Transforming matings into natural deduction proofs. In W. Bibel and R. Kowalski, editors, *Proceedings of the 5th CADE*, Les Arcs, France, 1980. Springer Verlag, Berlin, Germany, LNCS 87.

[3] W. W. Bledsoe. A Precondition Prover for Analogy. *Bio Systems*, 34:225–247, 1995. Also appeared as tech. report ATP 116, CS Department, U. Texas at Austin, USA.

[4] R. S. Boyer. *Locking: A Restriction of Resolution.* PhD thesis, U. Texas at Austin, USA, 1971.

[5] A. Bundy. The use of explicit plans to guide inductive proofs. In E. Lusk and R. Overbeek, editors, *Proceedings of the 9th CADE*, pages 111–120, Argonne, Illinois, USA, May1988. Springer Verlag, Berlin, Germany, LNCS 310.

[6] A. Bundy, A. Stevens, F. van Harmelen, A. Ireland, and A. Smaill. Rippling: A heuristic for guiding inductive proofs. *Artificial Intelligence*, 62:185–253, 1993.

[7] A. Bundy, F. van Harmelen, J. Hesketh, and A. Smaill. Experiments with proof plans for induction. Technical report, University of Edinburgh, Edinburgh, United Kingdom, 1988.

[8] A. Bundy, F. van Harmelen, C. Horn, and A. Smaill. The OYSTER-CLAM system. In M. E. Stickel, editor, *Proceedings of the 10th CADE*, pages 647–648, Kaiserslautern, Germany, 1990. Springer Verlag, Berlin, Germany, LNAI 449.

[9] R. L. Constable et al. *Implementing Mathematics with the Nuprl Proof Development System.* Prentice Hall, Englewood Cliffs, New Jersey, USA, 1986.

[10] F. Giunchiglia and P. Traverso. Program tactics and logic tactics. In F. Pfenning, editor, *Proceedings of LPAR*, pages 16–30, Kiev, Ukraine, 1994. Springer Verlag, Berlin, Germany, LNAI 822.

[11] M. Gordon, R. Milner, and C. Wadsworth. *Edinburgh LCF: A Mechanized Logic of Computation.* LNCS 78. Springer Verlag, Berlin, Germany, 1979.

[12] L. Hines. Str$\dot{+}$ve$\subseteq$: The Str$\dot{+}$ve-based subset prover. In M. Stickel, editor, *Proc. of the 10th CADE*, pages 193–206, Kaiserslautern, 1990. Springer Verlag, Berlin, Germany, LNAI 449.

[13] X. Huang, M. Kerber, and L. Cheikhrouhou. Adaptation of declaratively represented methods in proof planning. SEKI Report SR-95-12, Fachbereich Informatik, Universität des Saarlandes, Saarbrücken, Germany, 1995.

[14] X. Huang, M. Kerber, M. Kohlhase, E. Melis, D. Nesmith, J. Richts, and J. Siekmann. Ω-MKRP: A proof development environment. In A. Bundy, editor, *Proceedings of the 12th CADE*, pages 788–792, Nancy, 1994. Springer Verlag, Berlin, Germany, LNAI 814.

[15] X. Huang, M. Kerber, M. Kohlhase, and J. Richts. Adapting methods to novel tasks in proof planning. In B. Nebel and L. Dreschler-Fischer, editors, *KI-94: Advances in Artificial Intelligence – Proceedings of KI-94, 18th German Annual Conference on Artificial Intelligence*, pages 379–390, Saarbrücken, Germany, 1994. Springer Verlag, Berlin, Germany, LNAI 861.

[16] X. Huang, M. Kerber, J. Richts, and A. Sehn. Planning mathematical proofs with methods. *Journal of Information Processing and Cybernetics, EIK*, 30(5-6):277–291, 1994.

[17] A. Ireland. The use of planning critics in mechanizing inductive proofs. In A. Voronkov, editor, *Proceedings of LPAR*, pages 178–189, St. Petersburg, Russia, 1992. Springer Verlag, Berlin, Germany, LNAI 624.

[18] D. Loveland. *Automated Theorem Proving: A Logical Basis.* North-Holland, New York, USA, 1978.

[19] G. Pólya. *How to Solve It.* Princeton University Press, Princeton, New Jersey, USA, also as Penguin Book, 1990, London, United Kingdom, 1945.

[20] A. C. Sehn. DECLAME – eine deklarative Sprache zur Repräsentation von Methoden. Master's thesis, in German, Fachbereich Informatik, Universität des Saarlandes, October 1995. Also published as SEKI Working Paper SWP-95-02.

[21] K. VanLehn. Problem solving and cognitive skill acquisition. In M. I. Posner, editor, *Foundations of Cognitive Science*, chapter 14. MIT Press, Cambridge, Massachusetts, 1989.

[22] L. Wos, R. Overbeek, E. Lusk, and J. Boyle. *Automated Reasoning – Introduction and Applications.* Prentice Hall, Englewood Cliffs, New Jersey, USA, 1984.