CDCL-Based Abstract State Transition System for Coherent Logic*

Mladen Nikolić and Predrag Janičić

Faculty of Mathematics, University of Belgrade,
Belgrade, Studentski Trg 16, Serbia
{nikolic, janicic}@matf.bg.ac.rs

Abstract. We present a new, CDCL-based approach for automated theorem proving in coherent logic — an expressive semi-decidable fragment of first-order logic that provides potential for obtaining human readable and machine verifiable proofs. The approach is described by means of an abstract state transition system, inspired by existing transition systems for SAT and represents its faithful lifting to coherent logic. The presented transition system includes techniques from which CDCL SAT solvers benefited the most (backjumping and lemma learning), but also allows generation of human readable proofs. In contrast to other approaches to theorem proving in coherent logic, reasoning involved need not to be ground. We prove key properties of the system, primarily that the system yields a semidecision procedure for coherent logic. As a consequence, the semidecidability of another fragment of first order logic which is a proper superset of coherent logic is also proven.

Keywords: coherent logic, CDCL SAT solving, abstract state transition systems, machine verifiable proofs, readable proofs

1 Introduction

Coherent logic (CL) is a fragment of first-order logic that involves formulae of the form: $p_1(\vec{v}) \wedge \ldots \wedge p_n(\vec{v}) \Rightarrow \exists \vec{y}\, Q_1(\vec{v}, \vec{y}) \vee \ldots \vee \exists \vec{y}\, Q_m(\vec{v}, \vec{y})$ which are implicitly universally quantified and where $0 \leq n$, $0 \leq m$, $\vec{v}$ denotes a sequence of variables $v_1, v_2, \ldots, v_k$, p_i (for $1 \leq i \leq n$) denote atomic formulae (involving some of the variables from $\vec{v}$), $\vec{y}$ denotes a sequence of variables $y_1, y_2, \ldots, y_l$, and Q_j denote conjunctions of atomic formulae (involving some of the variables from $\vec{v}$ and $\vec{y}$). If $n = 0$, then the $p_1(\vec{v}) \wedge \ldots \wedge p_n(\vec{v})$ part is assumed to be $\top$, and if $m = 0$, then the $\exists \vec{y}\, Q_1(\vec{v}, \vec{y}) \vee \ldots \vee \exists \vec{y}\, Q_m(\vec{v}, \vec{y})$ part is assumed to be $\bot$. There are no function symbols with arity greater than 0.

CL was initially defined by Skolem and in recent years it gained new attention [2, 8, 4, 18]. It allows certain existential quantification, so it is more expressive than the resolution logic. In contrast to the resolution method, the conjecture

* This work was partially supported by the Serbian Ministry of Science grant 174021 and by SNF grant SCOPES IZ73Z0_127979/1.

being proved is kept unchanged and is directly proved (refutation, Skolemization and transformation to clausal form are not used). Hence, proofs in CL are natural and intuitive. In addition, reasoning is constructive and proof objects (verifiable by a proof assistant) can be easily obtained [2, 18]. The proof objects in CL also give readable proofs, which is significant for many applications (e.g., in formalizing mathematics and in education). A number of theories and theorems can be formulated directly and simply in CL.

CL is semi-decidable and there are several semi-decision procedures and corresponding theorem provers implemented for it and for similar logics [9, 18, 2, 15]. However, most (although not all) of them are rather simple forward-chaining procedures that can hardly tackle complex conjectures. For such tasks, CL needs more powerful proving engines. We believe that such engine can be based on the dominating CDCL (conflict-driven clause-learning based) approach for SAT solving [5]. A problem with CDCL-based systems is that, in general, they use clausal form, refutation, and Skolemization, so the obtained proofs are not readable (since they are not given in terms of the original signature). In this paper we present one such approach for CL, given in terms of abstract state transition systems, inspired by a transition system for SAT [10]. Our system is a generalization of the system for SAT and can be used as a base both for SAT solving and CL solving. An important distinguishing feature of our system is non-ground reasoning which promises large benefits in practice. Also, we take advantage of nature of CL and design our system so that readable proofs (for instance, in a natural language form or in the Isabelle/Isar form [19]) can be generated. The presented approach, is motivated by and built on the three strong pillars:

Suitability of CL. Coherent logic has a number of good features and is suitable for many automation tasks. It is very expressive and gives potential for obtaining both readable and machine verifiable proofs.

Practical advances in SAT. Over the last decade, a huge progress has been made in SAT solving: a number of high-level algorithmic and low-level implementation features have been developed, so modern SAT solvers can deal with industrial instances with hundreds of thousands of clauses. Our approach should enable the transfer of these advances to coherent logic.

Theoretical advances in SAT. SAT solvers have been described, precisely and suitably for rigorous mathematical analysis, in terms of abstract state transition systems. Their correctness has been proved, first informally [13, 10] and then formally (using a proof assistant) [11, 12]. These results helped a separation of different concepts used in SAT solvers (often intermixed in typical optimized implementations) and a deeper understanding of operation of SAT solvers. Ideas from transition systems for SAT were used in designing and describing our transition system and for proving its properties (it gives a decision procedure for SAT and a semidecision procedure for CL).

Overview of the paper. The rest of the paper is organized as follows: in Section 2 we give some relevant background information on CL, SAT, and transition systems for SAT; in Section 3 we present our abstract state transition system for CL and in Section 4 we outline its soundness and completeness proofs. In Section 5 we briefly present the mechanism for generating readable proofs based on the presented transition system. In Section 6 we discuss related work, and in Section 7 we draw final conclusions and discuss further work.

2 Background

Propositional logic, First order logic, SAT. We assume the standard notions of propositional and first-order logic (FOL). Propositional logic can be considered as a first-order logic theory with each propositional variable corresponding to a 0-arity predicate symbol. This convention enables considering both propositional logic and coherent logic within the context of first-order logic. For instance, the SAT problem can be defined in the context of FOL, in the following way: SAT *is a problem of deciding if it holds $\Gamma \models \bot$ (i.e., whether Γ is unsatisfiable), where Γ is a set of clauses over a signature $\mathcal{L}$ with no function symbols and only with predicate symbols of arity 0.* The SAT problem is decidable and is NP-complete.

Abstract State Transition Systems for SAT. The transition system (by Krstić and Goel [10]) given in Figure 1 and referred to as the *SAT system* hereafter, is used for checking if a set of propositional clauses is satisfiable. It is a terminating, sound and complete (under a certain restrictions) [12]: for any initial state, the system (subject to certain limitations to forget and restart) terminates and then, if C differs from *no_cflct*, then the current formula F is satisfiable (and so is the initial formula F_0) and the current trail M is its model and, otherwise, the current formula F is unsatisfiable (and so is the initial formula F_0).

Coherent logic. The definition of a coherent logic formula is given in Section 1. Coherent logic does not involve negation and the reasoning involved is intuitionistic. For an atomic formula A, $\neg A$ can be represented in the form $A \Rightarrow \bot$, but this translation is not applicable in a general case (for arbitrary formula). In order to reason about negated atomic formulae, for every predicate symbol p, typically a new predicate symbol $\overline{p}$ is introduced that stands for $\overline{p}$ and the following additional axioms are used [17]: $\forall \vec{x}(p(\vec{x}) \vee \overline{p}(\vec{x}))$, $\forall \vec{x}(p(\vec{x}) \wedge \overline{p}(\vec{x}) \Rightarrow \bot)$.

The validity problem in CL is a problem of deciding if it holds $\Gamma \models \Phi$, where Γ is a set of coherent formulae, Φ is a coherent formula, and $\models$ denotes the semantical consequence relation ($\Gamma \models \Phi$ holds if Φ is true in all non-empty Tarskian models for Γ). The validity problem in CL is undecidable, but semidecidable [3].

Any FOL formula can be translated into a CL formula with preserved validity [17]. However, this translation may rely on steps that involve classical logic.

Typically, along a proof of a CL formula, there are new (fresh) constants, *witnesses,* added to the current signature. The term *constant* is used both for the constant symbols from the initial signature and for the witnesses.

3 Abstract State Transition System for CL

In this section we present an abstract state transition system for CL, a base for a semi-decision procedure for CL. It extends and modifies the transition system

Decide:

$$\frac{l \in L \qquad l, \bar{l} \notin M}{M := M\ l^d}$$

UnitPropag:

$$\frac{l \vee l_1 \vee \ldots \vee l_k \in F \qquad \bar{l}_1, \ldots, \bar{l}_k \in M \qquad l, \bar{l} \notin M}{M := M\ l^i}$$

Conflict:

$$\frac{C = no\_cflct \qquad \bar{l}_1 \vee \ldots \vee \bar{l}_k \in F \qquad l_1, \ldots, l_k \in M}{C := \{l_1, \ldots, l_k\}}$$

Explain:

$$\frac{l \in C \qquad l \vee \bar{l}_1 \vee \ldots \vee \bar{l}_k \in F \qquad l_1, \ldots, l_k \prec l}{C := C \cup \{l_1, \ldots, l_k\} \setminus \{l\}}$$

Learn:

$$\frac{C = \{l_1, \ldots, l_k\} \qquad \bar{l}_1 \vee \ldots \vee \bar{l}_k \notin F}{F := F \cup \{\bar{l}_1 \vee \ldots \vee \bar{l}_k\}}$$

Backjump:

$$\frac{C = \{l, l_1, \ldots, l_k\} \qquad \bar{l} \vee \bar{l}_1 \vee \ldots \vee \bar{l}_k \in F \qquad \text{level } l > m \geq \text{level } l_i}{C := no\_cflct \qquad M := M^m\ \bar{l}^i}$$

Forget:

$$\frac{C = no\_cflct \qquad c \in F \qquad F \setminus c \models c}{F := F \setminus c}$$

Restart:

$$\frac{C = no\_cflct}{M := M^{[0]}}$$

Fig. 1. Transition system for SAT solving by Krstić and Goel ($l_i \prec l_j$ denotes that the literal l_i precedes l_j in M, l^d denotes a decision literal, l^i an implied literal, level l denotes the decision level of a literal l in M, and M^m denotes the prefix of M up to the level m)

for SAT given in Section 1. The two system share the same spirit and the rules from the SAT system have their counterparts, but typically in a more involved form. In the rest of the paper, the following form of an implicitly universally quantified formula will be considered:

$$\forall \vec{x}\ p_1(\vec{v}, \vec{x}) \wedge \ldots \wedge \forall \vec{x}\ p_n(\vec{v}, \vec{x}) \Rightarrow \exists \vec{y}\ q_1(\vec{v}, \vec{y}) \vee \ldots \vee \exists \vec{y}\ q_m(\vec{v}, \vec{y})$$

where atoms p_i involve some of the variables $\vec{v}$ and $\vec{x}$ and atoms q_i involve some of the variables from $\vec{v}$ and $\vec{y}$. In the rest of the text, by *coherent formula*, we mean a formula of this form. Note that there are two differences from the original coherent form. The first one is that q_i are atoms and not conjunctions of atoms. This restriction does not decrease the expressiveness, since there is a straightforward transformation from the original to this restricted form, since each conjunction can be attributed a new predicate symbol of an appropriate arity, which is then linked, by additional axioms, to that conjunction. Using these axioms, the introduced predicates can be eliminated from the object-level proofs. The second difference from the coherent form is that universal quantifiers may appear in the lefthand side of the formula. This extension can be easily avoided if necessary, but it has some beneficial consequences that we discuss later.

In the rest of the paper we, standardly, do not differentiate between the formulae equal up to renaming of variables. To simplify the presentation, the set of elements of a list L will also be denoted L and the empty list will also be denoted $\emptyset$. In addition, the set of quantified atoms in the conjunction $\mathcal{P}$ or disjunction $\mathcal{Q}$ will also be denoted $\mathcal{P}$ or $\mathcal{Q}$. Hence, for a coherent formula $\mathcal{P} \Rightarrow \mathcal{Q}$ where $\mathcal{P}$ is a conjunction of quantified atomic formulae and $\mathcal{Q}$ is a disjunction of quantified atomic formulae, $\mathcal{P}$ and $\mathcal{Q}$ can be also considered as sets. If $\mathcal{P} = \{p_1, \dots, p_n\}$ and $\mathcal{Q} = \{q_1, \dots, q_n\}$, $\forall\vec{x}\mathcal{P}$ will denote $\{\forall\vec{x}p_1, \dots, \forall\vec{x}p_n\}$ and $\exists\vec{y}\mathcal{Q}$ will mean $\{\exists\vec{y}q_1, \dots, \exists\vec{y}q_n\}$. Substitutions will be denoted λ, σ, σ', ...

Definition 1 (Signature and conjecture). *Let V be a countable set of variables and let $\mathcal{L} = (\Sigma^\infty, \Pi, ar)$ be a signature such that $\Sigma^\infty = \{c^i \mid i \in \mathbf{N} \setminus \{0\}\}$, where for each $i = 1, \dots$ it holds $ar(c^i) = 0$, and Π is a finite set of predicates with defined arities. Let no_cflct be a special symbol not appearing in the signature.*

Let there be given a coherent theory $\mathcal{T}$, i.e., a finite set of coherent axioms $\mathcal{AX}$, over a signature $(\Sigma_\mathcal{T}, \Pi, ar)$ where $\Sigma_\mathcal{T} = \{c^1, \dots, c^k\} \subseteq \Sigma^\infty$ and $k \geq 0$, and a conjecture $\forall\vec{x}\mathcal{H}^0(\vec{v},\vec{x}) \Rightarrow \mathcal{G}^0(\vec{v})$ (over the same signature), where $\mathcal{H}^0(\vec{v},\vec{x})$ is $h_1^0(\vec{v},\vec{x}) \wedge \dots \wedge h_m^0(\vec{v},\vec{x})$ and $\mathcal{G}^0(\vec{v})$ is $\exists\vec{y}\ g_1^0(\vec{v},\vec{y}) \vee \dots \vee \exists\vec{y}\ g_n^0(\vec{v},\vec{y})$ and h_i^0 and g_i^0 are atomic formulae. Let $\forall\vec{x}\mathcal{H}$ denote $\forall\vec{x}\mathcal{H}^0(\vec{v},\vec{x})\lambda$, let $\mathcal{G}$ denote $\mathcal{G}^0(\vec{v})\lambda$, for a ground substitution λ over $\vec{v}$ where all constants appearing in λ belong to $\Sigma^\infty \setminus \Sigma_\mathcal{T}$ and are pairwise distinct.

In our system, $\mathcal{H}$ will serve as an initial assumption and an element from $\mathcal{G}$ should be reached.

In SAT solving, the model is built incrementally by asserting literals. When both the positive and the corresponding negative literal are asserted the search branch is closed and the backtrack ensues. We define relevant elementary formulae that will take this role in our system.

Definition 2 (Quantified literal). Positive quantified literal *or a* quantified atom *is a ground atom or $p(\vec{v})$ or $\exists\vec{y}\ p(\vec{y})$ where p is a predicate symbol and $\exists\vec{y}\ p(\vec{y})$ is closed. Formulae $\forall\vec{x}\ p(\vec{v},\vec{x})$ and $\exists\vec{y}\ p(\vec{v},\vec{y})$ are* extended quantified atoms *or* eq-atoms. *A formula $(\forall\vec{x}\ p(\vec{v},\vec{x})) \Rightarrow \bot$ is a* negative quantified literal. *A* quantified literal *is a positive or negative quantified literal. An* extended quantified literal *or* eq-literal *is an extended quantified atom or a negative quantified literal. Instead of $l \Rightarrow \bot$, we may write $\overline{l}$.*

Example 1. In this and the following examples, constants $c^1, c^2, \dots$ will be referred to as $a, b, \dots$. We will assume $\Pi = \{p, q, r, s\}$ where $ar(p) = ar(q) = ar(r) = 2$ and $ar(s) = 1$. Formulae $p(a,b)$, $p(a,x)$, and $\exists y\ p(a,y)$ are quantified atoms. Formulae $\forall x\ p(a,x)$, $\forall x\ p(v,x)$, and $\exists y\ p(x,y)$ are extended quantified atoms (due to the presence of universal quantifiers in first two cases, and due to the presence of the free variable x in the third example). Formulae $\overline{p(a,b)}$, $\overline{p(a,x)}$, $\overline{p(x,y)}$, and $\overline{\forall x\ p(x,y)}$ are negative quantified literals.

Definition 3 (State). *A* state *is a 6-tuple $(\Sigma, \Gamma, M, \mathcal{C}_1, \mathcal{C}_2, \ell)$, where Σ is a finite list of elements from Σ^∞, Γ is a finite list of coherent formulae over V and*

(Σ, Π, ar), M is a list of (pairwise distinct) eq-literals (called a trail*) over V and (Σ, Π, ar), $\mathcal{C}_1 \Rightarrow \mathcal{C}_2$ is a formula called a* conflict implication*, and ℓ is the index of last introduced constant. An* initial state *is a state $S_0 = (\Sigma_0, \mathcal{AX}, \mathcal{H}, \mathcal{G}, \emptyset, \emptyset, n)$, where Σ_0 is the union of constants from $\mathcal{H} \cup \mathcal{G}$ and $\Sigma_{\mathcal{T}}$ (if Σ_0 is empty a constant from Σ^{∞} is added in it) and n is the maximal index of constants from Σ_0.*

Intuitively, the role of the state components is as follows: Σ stores the current set of constants, Γ stores the axioms and learnt lemmas, M stores the current set of inferred or assumed eq-literals. The inferred conclusions are logical consequences of the axioms along with assumed quantified literals. The formula $\mathcal{C}_1 \Rightarrow \mathcal{C}_2$ is used in a process called *conflict analysis*.

Definition 4 (Decision levels). *The elements of lists M and Σ are divided into* decision levels*. The elements of different decision levels are separated by the symbol $|$.*

The prefix of a list L that includes exactly the elements of the first m decision levels is denoted L^m.

For an element e, $L^{\frown m}e$ denotes a list obtained from L by inserting e at the end of the decision level m of L if $e \notin L$, and L if $e \in L$. If the number m is omitted, the last level is assumed.

We write $e \in^m L$ if e belongs to the m-th level of L. If $e_i \in^{m_i} L$ for $i = 1, \ldots, k$ where $k > 0$, and if $m = \max_i m_i$, then we write $\{e_1, \ldots, e_k\} \subseteq^m L$. If $k = 0$, we write $\emptyset \subseteq^0 L$.

We introduce relation denoting precedence of eq-literals in the trail ($\prec$).

Definition 5 (Relation $\prec$). *We write $l \prec l'$ in some state if it holds $M = M_1\ l\ M_2\ l' M_3$ in that state, where any of M_i (i=1,2,3) can be empty. For a set S, we write $S \prec l'$ if it holds $l \prec l'$ for each $l \in S$.*

Example 2. Let $M = [p(a,b), q(x,y), r(x,y)]$. Then $p(a,b) \prec q(x,y)$. It also holds $\{p(a,b), q(x,y)\} \prec r(x,y)$.

Next, we define the relations of entailment of eq-literals ($\Vdash$), and validity of eq-literal with respect to the current trail ($\uparrow$).

Definition 6 (Relations $\Vdash$ and $\uparrow$). *For eq-literals l and l' we write $l \Vdash l'$ if there is a substitution λ such that $l\lambda = l'$ and we write $l \Vdash \forall \vec{x} l'$ if $l \Vdash l'$ and we write $l \Vdash \exists \vec{y}\ l'$ if $l \Vdash\ l'[\vec{x} \mapsto \vec{t}]$ for some vector of variables and/or constants $\vec{t}$. We write $S \Vdash S'$ if there exist $l \in S$ and $l' \in S'$ such that $l \Vdash l'$. If some of these sets is singleton, we write its only element instead of it. We write $l \uparrow^m$ if there is a function m such that $m(l) \in M$ and $m(l) \Vdash l$.*

Example 3. It holds $p(x,y) \Vdash p(a,y)$, $p(x,y) \Vdash \forall x\, p(x,b)$, and $p(x,b) \Vdash \exists y \exists z\, p(y,z)$. Consequently, if it holds $p(x,b) \in M$, then it holds $\exists y \exists z\ p(y,z) \uparrow^m$ for any function m such that $m(\exists y \exists z\ p(y,z)) = p(x,b)$.

The following two definitions introduce the conflict between eq-literals ($\times$), and conflict of a formula with the current trail ($\downarrow$). Note that the term *conflict* is not used in the strict sense of contradiction, but it plays the same role the proper conflict plays in the SAT solving — it signals that backtracking is needed.

Definition 7 (Relation $\times$). *We write $p(\vec{x},\vec{t}) \times_\lambda \overline{\forall \vec{x}' p(\vec{x}',\vec{t'})}$ (or $\overline{\forall \vec{x}' p(\vec{x}',\vec{t'})} \times_\lambda p(\vec{x},\vec{t})$) if $\vec{t}\lambda = \vec{t'}\lambda$. We write $\exists y\ p(\vec{y},\vec{t}) \times_\lambda \overline{p(\vec{x}',\vec{t'})}$ (or $\overline{p(\vec{x}',\vec{t'})} \times_\lambda \exists y\ p(\vec{y},\vec{t})$) if $\vec{t}\lambda = \vec{t'}\lambda$. We write $l \times l'$ if it holds $l \times_\lambda l'$ for some λ.*

Example 4. It holds $p(a,b) \times \overline{p(x,y)}$, $p(x,y) \times \overline{\forall x\ p(x,b)}$, $p(x,b) \times \overline{\forall x\ p(x,b)}$, and $\exists x\ p(x,b) \times \overline{p(x,y)}$.

Definition 8 (Relation $\downarrow$). *We write $l \downarrow^m_\lambda$ if there is a function m such that $m(l) \in M$ and $l \times_\lambda m(l)$, and we write $l \downarrow$ if it holds $l \downarrow^m_\lambda$ for some m and λ. For a formula $\mathcal{P} \Rightarrow \mathcal{Q}$, a substitution λ, and a partial function $m : \mathcal{P} \cup \mathcal{Q} \to M$ we write $\mathcal{P} \Rightarrow \mathcal{Q} \downarrow^m_\lambda$ if for each $l \in \mathcal{P}$ it holds $m(l) \Vdash l\lambda$, and for each $l \in \mathcal{Q}$ it holds either $m(l) \times_\lambda l$ or $l\lambda \Vdash \mathcal{G}$ (in the latter case $m(l)$ is not defined). We write $\mathcal{P} \Rightarrow \mathcal{Q} \downarrow$ if it holds $\mathcal{P} \Rightarrow \mathcal{Q} \downarrow^m_\lambda$ for some m and λ. The function m is called the* conflict mapping *and the set $m(\mathcal{P} \cup \mathcal{Q})$ is called the* conflict set *for $\mathcal{P} \Rightarrow \mathcal{Q}$. We denote $\{l' \in \mathcal{P} \cup \mathcal{Q} \mid m(l') = l\}$ by $m^{-1}(l)$.*

Example 5. Let $\mathcal{G} = \exists x\ q(a,x) \vee \exists y\ q(y,b)$ and $M = [p(x,b), \overline{r(a,b)}, s(a)]$. It holds $(p(x,y) \Rightarrow r(x,y)) \downarrow^m_\lambda$ where $m(p(x,y)) = p(x,b)$ and $m(r(x,y)) = \overline{r(a,b)}$ and $\lambda = [x \mapsto a, y \mapsto b]$. It also holds $(p(x,y) \Rightarrow r(x,y) \vee q(x,b)) \downarrow^m_\lambda$ for the same m and λ (since $q(x,b)\lambda \Vdash \mathcal{G}$).

Negations of $\Vdash$, $\uparrow$, and $\downarrow$ are denoted $\nVdash$, $\not\uparrow$, and $\not\downarrow$ respectively.

For a fixed set of predicates, the set of all quantified atoms l over the signature Σ for which it holds $l \nVdash \mathcal{G}$ is denoted $\mathcal{A}(\Sigma)$. The restriction $l \nVdash \mathcal{G}$ on the set $\mathcal{A}(\Sigma)$ is imposed for technical convenience.

The conflict mapping can map several eq-atoms from a formula to the same quantified literal on the trail. We define the merging of these eq-atoms.

Definition 9 (Relation $\rightrightarrows_\lambda$). *Let $l_1 = \forall \vec{x}\ p(t_1, \dots, t_n)$ and $l_2 = \forall \vec{x}'\ p(t'_1, \dots, t'_n)$ such that $\vec{x} \cap \vec{x}' = \emptyset$. Let $J = \{i \mid t_i \notin \vec{x} \wedge t'_i \notin \vec{x}'\}$. We write $\{l_1, l_2\} \rightrightarrows_\lambda \forall \vec{x}''\ p(w_1, \dots, w_n)$ if:*

- *there is a most general unifier λ for all pairs (t_i, t'_i) $(i \in J)$;*
- *$u_i = t_i\lambda$ and $u'_i = t'_i\lambda$ are not constants if $i \notin J$;*
- *$w_i = u_i = u'_i$ for $i \in J$ and $w_i = u_i u'_i$ for $i \notin J$ where $u_i u'_i$ is a new variable and $u_i u'_i \in \vec{x}''$.*

For $n > 2$, we write $\{l_1, \dots, l_n\} \rightrightarrows_\lambda l$ if $\{l_1, l_2\} \rightrightarrows_\mu l'$ and $\{l', l_3, \dots, l_n\} \rightrightarrows_\nu l$ and $\lambda = \mu\nu$. If λ is not relevant it can be omitted. For eq-literals $\exists \vec{y}\ p(t_1, \dots, t_n)$ and $\exists \vec{y}'\ p(t'_1, \dots, t'_n)$, relation $\rightrightarrows_\lambda$ is defined by analogy.

Example 6. Denote $\mathcal{S} = \{\forall x \forall y\ \phi(x,x,y,y,u,v,w,c), \forall z\ \phi(z,z,z,z,z,z,t,t)\}$. It holds $\mathcal{S} \rightrightarrows_{[w \mapsto c, t \mapsto c]} \forall xz \forall yz \forall uz \forall vz\ \phi(xz,xz,yz,yz,uz,vz,c,c)$ where c is a constant. It also holds $\{\psi(u,u,v), \psi(v,w,u)\} \rightrightarrows_{[u \mapsto v, w \mapsto v]} \psi(v,v,v)$. It does not hold $\{\forall x\ \psi(x,u,u), \forall y \forall z\ \psi(y,z,c)\} \rightrightarrows_\lambda l$ for any l and any λ.

Definition 10 (CL transition system). *For a given (fixed) signature* $\mathcal{L} = (\Sigma^\infty, \Pi, ar)$*, a* CL transition system *is a system of rules*[1] *given in Figure 2. Each rule, when applicable, maps one state to another.*

$S \xrightarrow{r} S'$ *denotes that state* S' *can be obtained from state* S *by the rule* r*.* $S \rightarrow S'$ *denotes that* $S \xrightarrow{r} S'$ *holds for some rule* r*. A sequence of states* S_i *such that* $S_i \rightarrow S_{i+1}$ *is called a* chain*. A chain is* maximal *if it is finite and no rule is applicable in its last state or if it is infinite.*

Definition 11 (Final states). *An* accepting state *is a state* S *in which it holds* $\mathcal{C}_2 \neq \{no\_cflct\}$ *and* $\mathcal{C}_1 \Rightarrow \mathcal{C}_2 \downarrow^m$ *where* $m(\mathcal{C}_1 \cup \mathcal{C}_2) \subseteq \mathcal{H}$*. In that case, we write* $\mathcal{AX} \vdash_{CL} \forall \vec{x} \mathcal{H}^0(\vec{v}, \vec{x}) \Rightarrow \mathcal{G}^0(\vec{v})$*. A* rejecting state *is a state* S *for which there is no state* S' *such that* $S \rightarrow S'$ *and* S *is not an accepting state. A state is a* final state *if it is an accepting state or a rejecting state.*

Note that the condition $m(\mathcal{C}_1 \cup \mathcal{C}_2) \subseteq^0 M$ would suffice for the purpose of deciding validity. However, the stronger condition is suitable for purposes of object-level proof generation. We give informal explanations of the rules and some examples.

Decide: This rule makes an assumption. An assumption can be a quantified atom that is not redundant nor contradictory with respect to the current trail (for instance, if there is a quantified literal $s(y)$ on the trail, then the rule is not applicable for $s(c)$, $\exists x\ s(x)$). Within this step, in M and Σ it is denoted that a new level begins (that depends on the decision made).

Intro: This rule eliminates existential quantifiers and introduces new constant symbols as witnesses. For example, if $M = [p(a,b), \exists y\ q(x,y)]$, and $\Sigma = a, b$, the rule can insert a fresh constant symbol c into Σ and $q(a,c)$ (or $q(b,c)$) into M.

Unit propagate left/right: If the removal of an atom from a formula results in a conflict of that formula and the trail, that eq-literal can be propagated in positive or negative form depending on the side of the implication it belongs to. If $\overline{p(x,y)} \in M$, a formula $s(x) \Rightarrow \exists y\ p(x,y)$ can propagate $\overline{s(x)}$. If $\overline{p(a,y)} \in M$, $\overline{s(a)}$ can be propagated. Note that sometimes it is not the propagated eq-literal $\bar{l}\lambda$ that is logically implied, but the eq-literal $(l \Rightarrow \mathcal{G})\lambda$. Still, we propagate $\bar{l}\lambda$ for technical convenience. This does not jeopardize the soundness of our system nor the generation of object-level proofs. Also, note that the propagated eq-literal is inserted into M at the lowest level such that all objects it was derived from are below it.[2]

Branch end: If $\mathcal{P} \Rightarrow \mathcal{Q} \downarrow$ holds, then the current assumptions are either inconsistent or imply $\mathcal{G}$, so backtracking is needed. The process of *conflict analysis*[3] begins, which aims at finding a lemma that can be used to end subsequent branches that can be ended by derivation analogous to the current one.

[1] The explanations of the rules (we recommend the reader to read them in parallel with the definitions of the rules) and a detailed execution example follow.

[2] This corresponds to exhaustive unit propagation in SAT solving.

[3] The term (somewhat misleading in this context) comes from the SAT solving, where the goal is to reach a contradiction (i.e., a conflict). In CL, the goal is to reach a contradiction *or* the conclusions of a given target formula.

Decide:

$$\frac{l \in \mathcal{A}(\Sigma) \qquad l \not\uparrow \qquad l \not\downarrow}{M := M|l \qquad \Sigma := \Sigma|}$$

Intro:

$$\frac{\exists \vec{y}\; l \in M \qquad (\exists \vec{y}\; l)\lambda \in \mathcal{A}(\Sigma) \qquad l\lambda\lambda' \not\uparrow \ \text{ for any } \lambda'}{M := M^{\frown} l[y_1 \mapsto c^{\ell+1}, \ldots, y_k \mapsto c^{\ell+k}]\lambda \qquad \Sigma := \Sigma^{\frown} c^{\ell+1}, \ldots, c^{\ell+k} \qquad \ell := \ell + k}$$

Unit propagate left:

$$\frac{\mathcal{P} \cup \{l\} \Rightarrow \mathcal{Q} \in^{n_1} \Gamma \qquad \mathcal{P} \Rightarrow \mathcal{Q} \downharpoonleft_{\lambda}^{m} \qquad m(\mathcal{P} \cup \mathcal{Q}) \subseteq^{n_2} M \qquad \bar{l}\lambda \not\uparrow \qquad \bar{l}\lambda \not\downarrow}{M := M^{\frown \max(n_1,n_2)} \bar{l}\lambda}$$

Unit propagate right:

$$\frac{\mathcal{P} \Rightarrow \mathcal{Q} \cup \{l\} \in^{n_1} \Gamma \qquad \mathcal{P} \Rightarrow \mathcal{Q} \downharpoonleft_{\lambda}^{m} \qquad m(\mathcal{P} \cup \mathcal{Q})^{n_2} \subseteq M \qquad l\lambda \not\uparrow \qquad l\lambda \not\downarrow}{M := M^{\frown \max(n_1,n_2)} l\lambda}$$

Branch end:

$$\frac{\mathcal{C}_2 = \{no\_cflct\} \qquad \mathcal{P} \Rightarrow \mathcal{Q} \in \Gamma \qquad \mathcal{P} \Rightarrow \mathcal{Q} \downarrow}{\mathcal{C}_1 := \mathcal{P} \qquad \mathcal{C}_2 := \mathcal{Q}}$$

Explain left $\forall$:

$$\frac{\begin{array}{c} \mathcal{C}_1 \Rightarrow \mathcal{C}_2 \downharpoonleft^{m} \qquad l \in m(\mathcal{C}_1) \qquad \mathcal{S} = m^{-1}(l) \qquad \mathcal{S} \rightrightarrows \forall \vec{x} p(\vec{v}, \vec{x}) \\ \mathcal{P} \Rightarrow \mathcal{Q} \cup \{p(\vec{v}', \vec{x}')\} \in \Gamma \qquad \mathcal{P} \Rightarrow \mathcal{Q} \downharpoonleft^{m'} \qquad m'(\mathcal{P} \cup \mathcal{Q}) \prec l \qquad \overline{\forall \vec{x} p(\vec{v}, \vec{x})} \times_{\lambda} p(\vec{v}', \vec{x}') \end{array}}{\mathcal{C}_1 := (\forall \vec{x}' \mathcal{P} \cup (\mathcal{C}_1 \setminus \mathcal{S}))\lambda \qquad \mathcal{C}_2 := (\exists \vec{x}' \mathcal{Q} \cup \mathcal{C}_2)\lambda}$$

Explain left $\exists$:

$$\frac{\begin{array}{c} \mathcal{C}_1 \Rightarrow \mathcal{C}_2 \downharpoonleft^{m} \qquad l \in m(\mathcal{C}_1) \qquad \mathcal{S} = m^{-1}(l) \qquad \mathcal{S} \rightrightarrows_{\sigma} p(\vec{v}, \vec{x}) \\ \mathcal{P} \Rightarrow \mathcal{Q} \cup \{\exists \vec{x}' p(\vec{v}', \vec{x}')\} \in \Gamma \qquad \mathcal{P} \Rightarrow \mathcal{Q} \downharpoonleft^{m'} \qquad m'(\mathcal{P} \cup \mathcal{Q}) \prec l \qquad \overline{p(\vec{v}, \vec{x})} \times_{\lambda} \exists \vec{x}' p(\vec{v}', \vec{x}') \end{array}}{\mathcal{C}_1 := (\mathcal{P} \cup \forall \vec{x}(\mathcal{C}_1\sigma \setminus \mathcal{S}\sigma))\lambda \qquad \mathcal{C}_2 := (\mathcal{Q} \cup \exists \vec{x}(\mathcal{C}_2\sigma))\lambda}$$

Explain right $\forall$:

$$\frac{\begin{array}{c} \mathcal{C}_1 \Rightarrow \mathcal{C}_2 \downharpoonleft^{m} \qquad l \in m(\mathcal{C}_2) \qquad \mathcal{S} = m^{-1}(l) \qquad \mathcal{S} \rightrightarrows_{\sigma} p(\vec{v}, \vec{x}) \\ \{\forall \vec{x}' p(\vec{v}', \vec{x}')\} \cup \mathcal{P} \Rightarrow \mathcal{Q} \in \Gamma \qquad \mathcal{P} \Rightarrow \mathcal{Q} \downharpoonleft^{m'} \qquad m'(\mathcal{P} \cup \mathcal{Q}) \prec l \qquad p(\vec{v}, \vec{x}) \times_{\lambda} \overline{\forall \vec{x}' p(\vec{v}', \vec{x}')} \end{array}}{\mathcal{C}_1 := (\mathcal{P} \cup \forall \vec{x}(\mathcal{C}_1\sigma))\lambda \qquad \mathcal{C}_2 := (\mathcal{Q} \cup \exists \vec{x}(\mathcal{C}_2\sigma \setminus \mathcal{S}\sigma))\lambda}$$

Explain right $\exists$:

$$\frac{\begin{array}{c} \mathcal{C}_1 \Rightarrow \mathcal{C}_2 \downharpoonleft^{m} \qquad l \in m(\mathcal{C}_2) \qquad \mathcal{S} = m^{-1}(l) \qquad \mathcal{S} \rightrightarrows \exists \vec{x} p(\vec{v}, \vec{x}) \\ \{p(\vec{v}', \vec{x}')\} \cup \mathcal{P} \Rightarrow \mathcal{Q} \in \Gamma \qquad \mathcal{P} \Rightarrow \mathcal{Q} \downharpoonleft^{m'} \qquad m'(\mathcal{P} \cup \mathcal{Q}) \prec l \qquad \exists \vec{x} p(\vec{v}, \vec{x}) \times_{\lambda} \overline{p(\vec{v}', \vec{x}')} \end{array}}{\mathcal{C}_1 := (\forall \vec{x}' \mathcal{P} \cup \mathcal{C}_1)\lambda \qquad \mathcal{C}_2 := (\exists \vec{x}' \mathcal{Q} \cup (\mathcal{C}_2 \setminus \mathcal{S}))\lambda}$$

Learn:

$$\frac{\mathcal{C}_2 \neq \{no\_cflct\} \qquad \mathcal{C}_1 \Rightarrow \mathcal{C}_2 \notin \Gamma}{\Gamma := \Gamma^{\frown} \mathcal{C}_1 \Rightarrow \mathcal{C}_2}$$

Backjump:

$$\frac{\begin{array}{c} \mathcal{C}_1 \Rightarrow \mathcal{C}_2 \in \Gamma \qquad \mathcal{C}_1 \Rightarrow \mathcal{C}_2 \downharpoonleft^{m} \qquad l \in m(\mathcal{C}_1) \qquad \mathcal{S} = m^{-1}(l) \qquad \mathcal{C}_1 \setminus \mathcal{S} \Rightarrow \mathcal{C}_2 \downharpoonleft_{\lambda}^{m'} \\ m' \subseteq m \qquad m'(\mathcal{C}_1 \setminus \mathcal{S} \cup \mathcal{C}_2) \subseteq^{n} M \qquad l \in^{n'} M \qquad n \leq t < n' \qquad \mathcal{S}\lambda \rightrightarrows l' \end{array}}{M := M^{t \frown n} \bar{l}' \qquad \Sigma := \Sigma^{t} \qquad \mathcal{C}_1 := \emptyset \qquad \mathcal{C}_2 := \{no\_cflct\}}$$

Fig. 2. Abstract state transition system for CL

Explain left/right $\forall$/$\exists$: These rules perform conflict analysis by performing a kind of generalized resolution on the conflict implication and formulae from Γ that (could have) propagated quantified literals in the conflict set. The resolution can be described by following schematic rules, but in the Explain rules it is adjusted for resolving several literals at once when several eq-atoms from conflict implication correspond to the same quantified literal in the conflict set.

$$\frac{\mathcal{P} \Rightarrow \mathcal{Q} \cup \{\exists\vec{y}p(\vec{x},\vec{y})\} \quad \{p(\vec{x}',\vec{y}')\} \cup \mathcal{P}' \Rightarrow \mathcal{Q}'}{(\mathcal{P} \cup \forall\vec{y}'\mathcal{P}' \Rightarrow \mathcal{Q} \cup \exists\vec{y}'\mathcal{Q}')\lambda} \qquad \frac{\mathcal{P} \Rightarrow \mathcal{Q} \cup \{p(\vec{x},\vec{y})\} \quad \{\forall\vec{x}'p(\vec{x}',\vec{y}')\} \cup \mathcal{P}' \Rightarrow \mathcal{Q}'}{(\forall\vec{x}\mathcal{P} \cup \mathcal{P}' \Rightarrow \exists\vec{x}\mathcal{Q} \cup \mathcal{Q}')\sigma}$$

where λ is the most general unifier for $\vec{x}$ and $\vec{x}'$ and σ is the most general unifier for $\vec{y}$ and $\vec{y}'$. Notice that if length of $\vec{y}$ in the first case and $\vec{x}'$ in the second case is 0, the two rules are the same and in such case it is not important which one is used. Suppose a conflict implication is $p(x,y) \wedge q(x,y) \Rightarrow r(x,y)$ and that there is an axiom $s(x) \Rightarrow \exists y\ p(x,y)$. If the Explain left $\exists$ is applied to obtain a new conflict implication by resolving these two, $s(x) \wedge \forall y\ q(x,y) \Rightarrow \exists y\ r(x,y)$ is obtained. Now, suppose that the conflict implication is $p(v,z) \wedge \forall x\ q(x,v) \wedge \forall x\ q(v,x) \Rightarrow \exists y\ r(z,y)$, where both q eq-atoms in its lefthand side correspond to the same quantified atom in the conflict set, and that there is an axiom $p(x,y) \Rightarrow q(x,y)$. If Explain left $\forall$ is applied, since it holds $\{\forall x\ q(x,v), \forall x\ q(v,x)\} \rightrightarrows \forall u \forall w\ q(u,w)$, the new conflict implication is $(\forall v\ p(v,z) \wedge \forall x \forall v\ p(x,v)) \Rightarrow \exists y\ r(z,y)$.

Learn: In the conflict analysis process, an implication $\mathcal{C}_1 \Rightarrow \mathcal{C}_2$ is derived. Since it is a consequence of the axioms, it can be added to Γ as a learnt lemma.

Backjump: Since the conflict implication $\mathcal{C}_1 \Rightarrow \mathcal{C}_2$ is in conflict with M, some of the quantified literals have to be removed from the trail, so backjumping is performed. The level of the backjump is chosen so that only the quantified literal l from the top level in the conflict set is removed. Also, since other quantified literals from the conflict set are still present on the trail, a negative quantified literal can be derived from $\mathcal{C}_1 \Rightarrow \mathcal{C}_2$ that prevents l from appearing on the trail again. The last condition in the rule is concerned with the case when there are several quantified literals in $\mathcal{C}_1$ that correspond to l.

Example 7. Let us illustrate the operation of the CL transition system on the following (artificial) example. Let the axioms $\mathcal{AX}$ of $\mathcal{T}$ be (implicitly universally quantified) formulae: (Ax1) $p(x,y) \wedge q(x,y) \wedge r(x,y) \Rightarrow \bot$, (Ax2) $s(x) \Rightarrow \exists y\ q(x,y)$, (Ax3) $q(x,y) \Rightarrow r(x,y)$, and (Ax4) $s(x) \vee q(y,y)$, and the conjecture is $\forall z\ p(x,z) \Rightarrow \bot$.

The free variable of the conjecture is instantiated by fresh constant a: $\mathcal{H} = p(a,z)$ and $\mathcal{G} = \bot$. The initial state is $S_0 = (\{a\}, \mathcal{AX}, p(a,z), \emptyset, \emptyset, 1)$. The details of operation are given in the following table (note that the order in which the rules are applied is not fixed). Since the last state is an accepting state, the theorem has been proved.

Rule applied	Σ	$\Gamma \setminus \mathcal{AX}$ (lemmas)	M	$\mathcal{C}_1 \Rightarrow \mathcal{C}_2$
	a	$\emptyset$	$p(a,z)$	$\emptyset \Rightarrow \{no\_cflct\}$
Decide	$a\vert$	$\emptyset$	$p(a,z)\vert s(a)$	$\emptyset \Rightarrow \{no\_cflct\}$
U. p. right (Ax2)	$a\vert$	$\emptyset$	$p(a,z)\vert s(a), \exists y\ q(a,y)$	$\emptyset \Rightarrow \{no\_cflct\}$
Intro	$a\vert b$	$\emptyset$	$p(a,z)\vert s(a), \exists y\ q(a,y), q(a,b)$	$\emptyset \Rightarrow \{no\_cflct\}$
U. p. left (Ax1)	$a\vert b$	$\emptyset$	$p(a,z)\vert s(a), \exists y\ q(a,y), q(a,b), \overline{r(a,b)}$	$\emptyset \Rightarrow \{no\_cflct\}$
Branch end (Ax3)	$a\vert b$	$\emptyset$	$p(a,z)\vert s(a), \exists y\ q(a,y), q(a,b), \overline{r(a,b)}$	$q(x,y) \Rightarrow r(x,y)$
Ex. right $\forall/\exists$ (Ax1)	$a\vert b$	$\emptyset$	$p(a,z)\vert s(a), \exists y\ q(a,y), q(a,b), \overline{r(a,b)}$	$p(x,y) \wedge q(x,y) \Rightarrow \bot$
Ex. left $\exists$ (Ax2)	$a\vert b$	$\emptyset$	$p(a,z)\vert s(a), \exists y\ q(a,y), q(a,b), \overline{r(a,b)}$	$\forall y\ p(x,y) \wedge s(x) \Rightarrow \bot$
Learn	$a\vert b$	$\forall y\ p(x,y) \wedge s(x) \Rightarrow \bot$	$p(a,z)\vert s(a), \exists y\ q(a,y), q(a,b), \overline{r(a,b)}$	$\forall y\ p(x,y) \wedge s(x) \Rightarrow \bot$
Backjump	a	$\forall y\ p(x,y) \wedge s(x) \Rightarrow \bot$	$p(a,z), \overline{s(a)}$	$\emptyset \Rightarrow \{no\_cflct\}$
U. p. right (Ax4)	a	$\forall y\ p(x,y) \wedge s(x) \Rightarrow \bot$	$p(a,z), \overline{s(a)}, q(y,y)$	$\emptyset \Rightarrow \{no\_cflct\}$
U. p. left (Ax1)	a	$\forall y\ p(x,y) \wedge s(x) \Rightarrow \bot$	$p(a,z), \overline{s(a)}, q(y,y), \overline{r(a,a)}$	$\emptyset \Rightarrow \{no\_cflct\}$
Branch end (Ax3)	a	$\forall y\ p(x,y) \wedge s(x) \Rightarrow \bot$	$p(a,z), \overline{s(a)}, q(y,y), \overline{r(a,a)}$	$q(x,y) \Rightarrow r(x,y)$
Ex. right $\forall/\exists$ (Ax1)	a	$\forall y\ p(x,y) \wedge s(x) \Rightarrow \bot$	$p(a,z), \overline{s(a)}, q(y,y), \overline{r(a,a)}$	$p(x,y) \wedge q(x,y) \Rightarrow \bot$
Ex. left (Ax4)	a	$\forall y\ p(x,y) \wedge s(x) \Rightarrow \bot$	$p(a,z), \overline{s(a)}, q(y,y), \overline{r(a,a)}$	$p(x,x) \Rightarrow s(z)$
Ex. right (lemma)	a	$\forall y\ p(x,y) \wedge s(x) \Rightarrow \bot$	$p(a,z), \overline{s(a)}, q(y,y), \overline{r(a,a)}$	$p(x,x) \wedge \forall u\ p(z,u) \Rightarrow \bot$

As already noted, the fragment we are working with is syntactically broader than CL due to the presence of universal quantifiers in the lefthand side of the formulae. This extension allows more expressive lemma learning mechanism, compliant with the use of (implicitly) universally quantified literals on the trail. If one wants to stay within the original CL fragment, three conditions should be fulfilled. The rule Decide should insert only closed quantified atoms on the trail. The Explain rules should choose the literal l from the conflict set such that all other literals from the set precede it on the trail and should resolve the conflict implication with an axiom (or a learnt lemma) that was used to propagate (or derive in the backjump) the literal l. Finally, if there are universal quantifiers in the left side of the conflict implication, the Explain rules should be applied until these quantifiers are removed.

4 Properties of CL Transition Systems

In this section we state the properties of the proposed system. Both soundness and completeness are proven with respect to non-empty Tarskian models of the axiom set. The relation $\models$ denotes a logical consequence. Full proofs of the theorems are available in the appendix.[4]

The soundness of the transition system given in Figure 2 is proven by showing that if an accepting state is reached for a coherent formula Φ and a coherent theory $\mathcal{AX}$, then Φ is a logical consequence of $\mathcal{AX}$.

Theorem 1 (Soundness). *If it holds* $\mathcal{AX} \vdash_{CL} \forall\vec{x}\mathcal{H}^0(\vec{v},\vec{x}) \Rightarrow \mathcal{G}^0(\vec{v})$*, then it holds* $\mathcal{AX} \models \forall\vec{x}\mathcal{H}^0(\vec{v},\vec{x}) \Rightarrow \mathcal{G}^0(\vec{v})$.

Proof outline. Since the accepting state can be reached, there is a conflict implication $\mathcal{C}_1 \Rightarrow \mathcal{C}_2$ and its corresponding resolution proof (in the sense of generalized resolution defined by the Explain rules). Resolution steps defined by the Explain rules are sound, and they involve only the axioms and previously learnt lemmas (derived again from the axioms), so $\mathcal{C}_1 \Rightarrow \mathcal{C}_2$ is a logical consequence of $\mathcal{AX}$. By the definition of an accepting state, the conflict set $\mathcal{S}$ for $\mathcal{C}_1 \Rightarrow \mathcal{C}_2$ is a subset of $\mathcal{H}$. Moreover, by the definition of the conflict set, $\mathcal{S}$ contains positive quantified literals that satisfy all conjuncts from $\mathcal{C}_1\lambda$ for some λ. Also, each disjunct from $\mathcal{C}_2$ either implies $\mathcal{G}$ or corresponds to a negative quantified literal from $\mathcal{S}$. Since $\mathcal{S} \subseteq \mathcal{H}$ and $\mathcal{H}$ has no negative quantified literals, in $\mathcal{C}_2$ there can be only disjuncts that imply $\mathcal{G}$. Therefore, $\mathcal{C}_1 \Rightarrow \mathcal{G}$ is a logical consequence of $\mathcal{AX}$, so $(\mathcal{C}_1 \Rightarrow \mathcal{G})\lambda$ and (since $\mathcal{G}$ is closed) $\mathcal{C}_1\lambda \Rightarrow \mathcal{G}$ are logical consequences of $\mathcal{AX}$. If $\mathcal{H}$ is true in some model of $\mathcal{AX}$, then $\mathcal{C}_1\lambda$ is true in that model, and consequently $\mathcal{G}$, too.

There are no guarantees that an arbitrary order of rule applications for a valid formula leads to the accepting state. In order to ensure this property, that we call *strong completeness*, we introduce the following restrictions to our system.

[4] The appendix is available online from `http://argo.matf.bg.ac.rs/cdclcl.pdf`

We extend the state with a list of numbers $\varDelta$ and a number δ. So, a state is a 8-tuple $(\varSigma, \varGamma, M, \mathcal{C}_1, \mathcal{C}_2, \ell, \varDelta, \delta)$. In an initial state S_0, it holds $\delta_0 = \ell_0$ and $\varDelta_0 = \delta_0$. Also, we add the *limiter rule*:

Limiter:

$$\frac{\text{No other rules applicable} \qquad \delta < max\{i \mid c^i \in \varSigma\}}{\delta := \delta + 1}$$

The indices of constants in the quantified atom l in Decide and Intro rule have to be less than or equal to δ. The same holds for substitution λ in Backjump and Unit propagate. Effects of the Decide rule are extended by $\varDelta := \varDelta|\ \delta$ and of the Backjump rule by $\varDelta := \varDelta^m$ and $\delta := \delta'$ where δ' is the last element of $\varDelta^m$. Although weaker restriction can be made, we restrict the Explain rules to choose the literal l from the conflict set such that all other literals from the set precede it on the trail, and to resolve the conflict implication with an axiom (or a learnt lemma) that was used to propagate (or derive in the backjump) the literal l. These formulae (called reason clauses in SAT solving) can be easily found if the record is kept when Unit propagation and Backjump are applied, as is the common practice in SAT solving. Note that the new system is a restriction of the original one, so the soundness arguments hold for the new system, too. The strong completeness can be proven — that for a valid formula, any order of rule applications (compliant with the imposed restrictions) will reach an accepting state.

Theorem 2 (Strong completeness). *If it holds $\mathcal{AX} \models \forall \vec{x} \mathcal{H}^0(\vec{v}, \vec{x}) \Rightarrow \mathcal{G}^0(\vec{v})$, then in each maximal chain $S_0 \rightarrow \ldots$ there exists an accepting state.*

Proof outline. Suppose that there is no accepting state in the chain. Then, the chain is either infinite or ends in a state in which no rule is applicable. Let L be the set of all quantified atoms that are permanently kept on the trail after some state in the chain. Consider a ground model $\mathcal{M}$ in which a ground atom l is true if $L \Vdash l$. If the premises of an axiom are true in $\mathcal{M}$, it can be shown that its conclusions are also true in $\mathcal{M}$. So, $\mathcal{M}$ is a model for $\mathcal{AX}$. $\mathcal{H}$ is trivially true in $\mathcal{M}$. Let us suppose that in some state S it holds $g \uparrow$ for some $g \in \mathcal{G}$. This indicates a branch end, and it can be shown that a backjump follows, after which $g \not\uparrow$ holds. Hence $\mathcal{G}$ is not true in $\mathcal{M}$. This shows that $\mathcal{M}$ is not a model for $(\forall * \mathcal{H}) \Rightarrow \mathcal{G}$ even though it is a model for $\mathcal{AX}$ which is a contradiction with the assumptions of the statement.

5 Generation of Readable Proofs

CDCL-based systems typically provide resolution refutation proofs that are not readable because transformation to clausal form, refutation and Skolemization are used. These proofs can, in principle, be transformed to forward chaining proofs, but these proofs would be hardly readable (because of the transformed axioms and function symbols non-existent in the original theory) and would not resemble textbook proofs (e.g., in geometry). This is avoided in our system, enabling generation of forward chaining proofs that can serve for simple building of readable proofs (for instance, in a natural language form or in the Isabelle/Isar form). We define a coherent forward chaining proof system (inspired by proof system given in [17]) as follows. The axioms are $\Gamma, \bot \vdash \Phi$ and $\Gamma, \phi \vdash \Phi$ if $\phi \Vdash \Phi$. The rules are:

$$\frac{\Gamma, A, A \Rightarrow B, B \vdash \Phi}{\Gamma, A, A \Rightarrow B \vdash \Phi} \Rightarrow \qquad \frac{\Gamma, A \vdash \Phi \quad \Gamma, B \vdash \Phi}{\Gamma, A \vee B \vdash \Phi} \vee \qquad \frac{\Gamma, A, A\sigma \vdash \Phi}{\Gamma, A \vdash \Phi}\ \mathit{Inst}$$

$$\frac{\Gamma, p(\vec{x}), \forall \vec{x}\ p(\vec{x}) \vdash \Phi}{\Gamma, p(\vec{x}) \vdash \Phi} \forall \qquad \frac{\Gamma, \exists \vec{y}\ p(\vec{x}, \vec{y}), p(\vec{a}, \vec{c}) \vdash \Phi}{\Gamma, \exists \vec{y}\ p(\vec{x}, \vec{y}) \vdash \Phi} \exists \qquad \frac{\Gamma, p(\vec{a}), p(\vec{x}) \vdash \Phi}{\Gamma, p(\vec{a}) \vdash \Phi}\ \mathit{Eigen}$$

where in $\vee$ rule A and B share no free variables, in Inst, σ is an arbitrary substitution over variables and constants from Σ^{∞}, in $\exists$ rule $\vec{a}$ consists of constants appearing in Γ and $\exists \vec{y}\ p(\vec{x}, \vec{y})$ and $\vec{c}$ consists of fresh constants, and Eigen can be applied to constants from $\vec{a}$ only in one branch of the proof and those constants must have been introduced by Inst as fresh constants at the step in which they first appear (this is a kind of eigenvariable condition). For a formula $\forall \vec{x} \mathcal{H}^0(\vec{v}, \vec{x}) \Rightarrow \mathcal{G}^0(\vec{v})$ the coherent forward chaining proof is a derivation tree for $\mathcal{AX}, \mathcal{H} \vdash \mathcal{G}$.

Theorem 3. *If $\mathcal{AX} \vdash_{CL} \forall \vec{x} \mathcal{H}^0(\vec{v}, \vec{x}) \Rightarrow \mathcal{G}^0(\vec{v})$, there exists a coherent forward chaining proof for $\forall \vec{x} \mathcal{H}^0(\vec{v}, \vec{x}) \Rightarrow \mathcal{G}^0(\vec{v})$.*

Proof outline. Since the accepting state can be reached, there is a conflict implication $\mathcal{C}_1 \Rightarrow \mathcal{C}_2$ and its resolution proof built from the axioms. A proof of the conjecture can be generated recursively from this resolution proof. Let resolving an eq-literal from the lefthand side of some $\mathcal{F}_1$ and an eq-literal from the righthand side of some $\mathcal{F}_2$ yield a conflict implication $\mathcal{C}_1 \Rightarrow \mathcal{C}_2$. If Pr_1 and Pr_2 are forward chaining proofs for $\mathcal{F}_1$ and $\mathcal{F}_2$, then the proof of the conjecture is constructed (roughly) by replacing non-contradiction leafs of Pr_1 by Pr_2 (with appropriate renaming of fresh constants and free variables in Pr_2).

A generated proof need not be axiom-level, but can involve learnt lemmas with their proofs generated separately.

Example 8. We present the resolution tree for last derived $\mathcal{C}_1 \Rightarrow \mathcal{C}_2$, corresponding to the example of system execution given in Example 7.

$$\dfrac{\dfrac{s(x) \vee q(y,y) \quad \dfrac{q(x,y) \Rightarrow r(x,y) \quad p(x,y) \wedge q(x,y) \wedge r(x,y) \Rightarrow \bot}{p(x,y) \wedge q(x,y) \Rightarrow \bot}}{p(x,x) \Rightarrow s(z)} \qquad \dfrac{s(x) \Rightarrow \exists y\ q(x,y) \quad \dfrac{q(x,y) \Rightarrow r(x,y) \quad p(x,y) \wedge q(x,y) \wedge r(x,y) \Rightarrow \bot}{p(x,y) \wedge q(x,y) \Rightarrow \bot}}{\forall y\ p(x,y) \wedge s(x) \Rightarrow \bot}}{p(x,x) \wedge \forall u\ p(z,u) \Rightarrow \bot}$$

To make the notation more readable, in forward chaining proofs, we do not write the hole context, but only the last derived fact. The forward chaining proofs for $p(x,x) \Rightarrow s(z)$ and $\forall y\ p(x,y) \wedge s(x) \Rightarrow \bot$ are:

$$\dfrac{\dfrac{s(b)\vdash s(b)}{s(x)\vdash s(b)}\,Inst \qquad \dfrac{\dfrac{\dfrac{\dfrac{\bot\vdash s(b)}{r(a,a)\vdash s(b)}\Rightarrow(Ax1)}{q(a,a)\vdash s(b)}\,Inst}{r(y,y)\vdash s(b)}\,Inst}{q(y,y)\vdash s(b)}\Rightarrow(Ax3)}{\mathcal{AX}, p(a,a)\vdash s(b)}\,\vee \qquad\qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\bot\vdash\bot}{p(a,b)\vdash\bot}\Rightarrow(Ax1)}{r(a,b)\vdash\bot}\,Inst}{q(a,b)\vdash\bot}\Rightarrow(Ax3)}{\exists y\ q(a,y)\vdash\bot}\,\exists}{\mathcal{AX}, p(a,y), s(a)\vdash\bot}\Rightarrow(Ax2)$$

If we denote the first one by Pr_1 and the second one by Pr_2, and apply the rules applied in Pr_1 starting from the root $\mathcal{AX}, p(a,z) \vdash \bot$, we obtain Pr'_1. Then starting from each leaf of Pr'_1 that did not end in contradiction, we can apply all the rules applied in Pr_2. The obtained proof is the proof for conjecture:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\bot\vdash\bot}{p(a,b)\vdash\bot}\Rightarrow(Ax1)}{r(a,b)\vdash\bot}\,Inst}{q(a,b)\vdash\bot}\Rightarrow(Ax3)}{\exists y\ q(a,y)\vdash\bot}\,\exists}{s(a)\vdash\bot}\Rightarrow(Ax2)}{s(x)\vdash\bot}\,Inst \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\bot\vdash\bot}{p(a,a)\vdash\bot}\Rightarrow(Ax1)}{r(a,a)\vdash\bot}\,Inst}{q(a,a)\vdash\bot}\,Inst}{r(y,y)\vdash\bot}\,Inst}{q(y,y)\vdash\bot}\Rightarrow(Ax3)}{\mathcal{AX}, p(a,z)\vdash\bot}\,\vee$$

A corresponding readable (Isar-style) proof would be as follows: *Assume* $\forall z\ p(a,z)$. *With (Ax4), it holds* $\forall x\ s(x)$ *or* $\forall y\ q(y,y)$. *Assume* $\forall x\ s(x)$. *From* $\forall x\ s(x)$, *it holds* $s(a)$. *With (Ax2), it holds* $\exists y\ q(a,y)$. *From* $\exists y\ q(a,y)$, *obtain* b *such that* $q(a,b)$. *With (Ax3), it holds* $r(a,b)$. *From* $\forall z\ p(a,z)$, *it holds* $p(a,b)$. *With (Ax1), this leads to a contradiction. Assume* $\forall y\ q(y,y)$. *With (Ax3), it holds* $\forall y\ r(y,y)$. *From* $\forall y\ q(y,y)$, *it holds* $q(a,a)$. *From* $\forall y\ r(y,y)$, *it holds* $r(a,a)$. *From* $\forall z\ p(a,z)$, *it holds* $p(a,a)$. *With (Ax1), this leads to a contradiction. All the branches are closed and the conjecture has been proven.*

6 Related Work

There are several proving procedures for coherent logic and similar fragments of FOL, and several corresponding automated theorem provers. To our knowledge, the first CL automated theorem prover was developed in Prolog by Janičić and Kordić [9] and was used for one axiomatization of Euclidean geometry. This prover was later reimplemented in C++ to give a more efficient and generic theorem prover ArgoCLP that produces both natural language proofs and object level proofs in the Isabelle form [18]. Bezem and Coquand developed in Prolog a sound and complete CL prover [2] based on breadth-first search that generates proof objects in Coq. Berghofer and Bezem developed an internal prover for CL in ML to be used within the system Isabelle [19]. Neither of these provers uses backjumps or lemma learning. De Nivelle implemented a theorem prover for logic close to coherent logic, that uses a mechanism for learning lemmas of somewhat restricted form [15]. All of these systems perform only ground reasoning.

Our work is also related to research focused on CDCL-based SAT solvers. Various modifications to the original DPLL procedure have been proposed, both on the high, logical level, and on the lower, algorithmic and implementation level [5]. Modern SAT solvers have been recently described (with some implementation features omitted) via abstract state transition systems [13, 10]. These systems provided a solid ground for rigorous analysis of the SAT solvers and their correctness was formally proved within the system Isabelle [12]. On the operational, practical level, our CL system enables the transfer of SAT algorithms and heuristics (that had a great impact on SAT solving) to CL in some form.

Our system builds on the SAT system presented in Section 2. The differences are primarily due to the first order nature and the form of coherent logic. Use of existential quantifiers results in use of Intro rule which is not present in SAT. Also, dealing with first order formulae results in more complex rules due to the use of substitutions and quantifiers. At this moment we do not include Forget and Restart rules, which can be trivially added, but with them an additional care has to be taken not to jeopardize completeness.

Our work is also related to work on *effectively propositional logic*, also known as EPR, or as the Bernays-Schönfinkel fragment of first-order logic [16].

Another lifting of DPLL procedure, to the clausal fragment of first order logic, is the Model evolution calculus [1]. There are several differences between this system and ours. The first is the underlying logic itself. Working in CL, one can avoid transformation to clausal form when working with coherent theories (like geometry). In our system, the refutation is avoided and forward proofs are used. Skolemization is avoided and existential quantifiers are used. These properties enable generation of readable proofs, close to proofs from mathematical textbooks. Also, in Model evolution calculus, backjump is not treated as a part of the calculus, but as an implementation technique [1].

7 Conclusions and Future Work

In this paper we presented an abstract state transition system for proving validity in coherent logic, but also in a somewhat broader fragment of first order logic. The system is sound and complete: any formula proved by the system is indeed a theorem, and for any input theorem, the system can and will prove it. This also proves the semidecidability of the defined extension of coherent logic. The system is based on a transition system for CDCL-based SAT solving. In contrast to other coherent theorem provers, the reasoning need not to be ground. An important property is that the system allows the generation of formal and human readable forward chaining proofs and we showed how they can be generated.

We are currently developing the implementation that faithfully matches the presented system and expect it to perform well compared to the existing provers. Heuristics that guide applications of the rules should be devised to improve the performance, hopefully in the spirit of heuristics for SAT [14]. Also, the generation of formal (in Isabelle/Isar) and readable object-level proofs will be implemented. We are planning to use the prover for a range of applications, including applications in formalization of mathematics, education, and program synthesis.

References

1. P. Baumgartner and C. Tinelli. The Model Evolution Calculus as a First-Order DPLL Method. *Artificial Intelligence*, 172(4-5), 2008.
2. M. Bezem and T. Coquand. Automating coherent logic. *LPAR 2005*, LNCS 3835, Springer 2005.
3. M. Bezem. On the Undecidability of Coherent Logic. *Processes, Terms and Cycles*, 2005.
4. M. Bezem and D. Hendriks. On the Mechanization of the Proof of Hessenberg's Theorem in Coherent Logic. *J. of Automated Reasoning*, 40(1), 2008.
5. A. Biere, M. Heule, H. van Maaren, and T. Walsh, editors. *Handbook of Satisfiability*, IOS Press, 2009.
6. M. Davis and H. Putnam. A Computing Procedure for Quantification Theory. *J. of ACM*, 7(3), 1960.
7. M. Davis, G. Logemann, and D. Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5(7), 1962.
8. J. Fisher and M. Bezem. Skolem machines and geometric logic. *International Colloquium on Theoretical Aspects of Computing 2007*, LNCS 4711, Springer, 2007.
9. P. Janičić and S. Kordić. EUCLID — the geometry theorem prover. *FILOMAT*, 9(3), 1995.
10. S. Krstić and A. Goel. Architecting solvers for sat modulo theories: Nelson-oppen with DPLL. In *FROCOS 2007*, LNCS 4720, Springer, 2007.
11. F. Marić. Formalization and Implementation of Modern SAT Solvers. *J. of Automated Reasoning*, 43(1), 2009.
12. F. Marić and P. Janičić. Formalization of Abstract State Transition Systems for SAT. *Logical Methods in Computer Science*, 7(3), 2011.
13. R. Nieuwenhuis, A. Oliveras, and C. Tinelli. Solving SAT and SAT modulo theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(T). *J. of ACM*, 53(6), 2006.
14. M. Nikolić, F. Marić, and P. Janicić. Instance-Based Selection of Policies for SAT Solvers, *SAT 2009*, LNCS 5584, Springer, 2009.
15. H. de Nivelle, J. Meng: Geometric Resolution: A Proof Procedure Based on Finite Model Search. *IJCAR 2006*, LNCS 4130, Springer, 2006.
16. R. Piskač, L. de Moura, and N. Bjorner. Deciding effectively propositional logic using DPLL and substitution sets. *J. of Automated Reasoning*, 44, 2010.
17. A. Polonsky. Proofs, Types, and Lambda Calculus PhD thesis, University of Bergen, 2010.
18. S. Stojanović, V. Pavlović, P. Janičić. Automated Generation of Formal and Readable Proofs in Geometry Using Coherent Logic. *ADG 2010*, LNCS 6877, Springer, 2011.
19. M. Wenzel. Isar - A Generic Interpretative Approach to Readable Formal Proof Documents, *TPHOLs 1999*, LNCS 1690, Springer, 2002.