A Resolution Prover for Coalition Logic

Cláudia Nalon  
Department of Computer Science  
University of Brasília (Brazil)  
nalon@unb.br

Lan Zhang  
Information School  
Capital University of Economics and Business (China)  
lan@cueb.edu.cn

Clare Dixon  
Ullrich Hustadt  
Department of Computer Science  
University of Liverpool (UK)  
{CLDixon,U.Hustadt}@liverpool.ac.uk

We present a prototype tool for automated reasoning for Coalition Logic, a non-normal modal logic that can be used for reasoning about cooperative agency. The theorem prover CLProver is based on recent work on a resolution-based calculus for Coalition Logic that operates on coalition problems, a normal form for Coalition Logic. We provide an overview of coalition problems and of the resolution-based calculus for Coalition Logic. We then give details of the implementation of CLProver and present the results for a comparison with an existing tableau-based solver.

1 Introduction

Coalition Logic CL is a formalism intended to describe the ability of groups of agents to achieve an outcome in a strategic game [14]. CL is a multi-modal logic with modal operators of the form $[\mathcal{A}]$, where $\mathcal{A}$ is a set of agents. The formula $[\mathcal{A}]\varphi$ reads as the coalition $\mathcal{A}$ has a strategy to achieve $\varphi$, where $\varphi$ is a formula. We note that CL is a non-normal modal logic, as the schema that represents additivity, $[\mathcal{A}]\varphi \land [\mathcal{A}]\psi \Rightarrow [\mathcal{A}](\varphi \land \psi)$, is not valid. However, monotonicity, $[\mathcal{A}](\varphi \land \psi) \Rightarrow [\mathcal{A}][\mathcal{A}]\varphi \land [\mathcal{A}]\psi$, holds.

Coalition Logic is equivalent to the next-time fragment of Alternating-Time Temporal Logic (ATL) [1, 5], where $[\mathcal{A}]\varphi$ translates into $\langle\langle \mathcal{A} \rangle \rangle \varphi$ (read as the coalition $\mathcal{A}$ can ensure $\varphi$ at the next moment in time). The satisfiability problems for ATL and CL are EXPTIME-complete [16] and PSPACE-complete [14], respectively. Proof methods for these logics include, for instance, tableau-based methods for ATL [16, 6] and a tableau-based method for CL [8].

In order to make the paper self-contained, we present here the resolution-based calculus for CL, RES$_{CL}$ [12]. As to the best of our knowledge, there are no other resolution-based methods for either ATL or CL. Providing such a method for CL gives the user a choice of proof methods. Several comparisons of tableau algorithms and resolution methods [10, 7] indicate that there is no overall best approach: for some classes of formulae tableau algorithms perform better whilst on others resolution performs better. So, with a choice of different provers, for the best result, the user could run several in parallel or the one most likely to succeed depending on the type of the input formulae. RES$_{CL}$ is sound, complete, and terminating as shown in [12].

The paper is organised as follows. In the next section, we present the syntax, axiomatisation, and semantics of CL. In Section 3, we introduce the resolution-based method for CL, the main results, and provide a small example. In Section 4, we introduce the theorem-prover for CL. We give details of the implementation and discuss the results for a comparison with an existing tool. Conclusions and future work are given in Section 5.
2 Coalition Logic

As in [6], we define $\Sigma \subseteq \mathbb{N}$ to be a finite, non-empty set of agents. A coalition $\mathcal{A}$ is a subset of $\Sigma$. Formulae in CL are constructed from propositional symbols ($\Pi = \{ p, q, r, \ldots, p_1, q_1, r_1, \ldots \}$) and constants (true, false), together with Boolean operators ($\neg$, for negation, and $\land$, for conjunction) and coalition modalities. Formulae whose main operator is classical are built in the usual way. A coalition modality is either of the form $[\mathcal{A}]\varphi$ or $(\mathcal{A})\varphi$, where $\varphi$ is a well-formed CL formula. The coalition operator $[\mathcal{A}]$ is the dual of $(\mathcal{A})$, that is, $(\mathcal{A})\varphi$ is an abbreviation for $\neg [\mathcal{A}] \neg \varphi$, for every coalition $\mathcal{A}$ and formula $\varphi$. We denote by WFF$_{CL}$ the set of CL well-formed formulae. Parentheses will be omitted if the reading is not ambiguous. We also omit the curly brackets within modalities. For instance, we write $[\mathcal{A}] \varphi$ instead of $[(\{1, 2\})] \varphi$.

A literal is either $p$ or $\neg p$, for $p \in \Pi$. For a literal $l$ of the form $\neg p$, where $p$ is a propositional symbol, $\neg l$ denotes $p$; for a literal $l$ of the form $p$, $\neg l$ denotes $\neg p$. The literals $l$ and $\neg l$ are called complementary literals. We assume that literals are in simplified form, that is, $\neg \neg l$ is assumed to be $l$.

A positive coalition formula (resp. negative coalition formula) is a formula of the form $[\mathcal{A}]\varphi$ (resp. $(\mathcal{A})\varphi$), where $\varphi \in$ WFF$_{CL}$. A coalition formula is either a positive or a negative coalition formula.

Coalition logic can be axiomatised by the following schemata (where $\mathcal{A}, \mathcal{A}'$ are coalitions and $\varphi, \varphi_1, \varphi_2$ are well-formed formulae) [14]:

\[ \begin{align*}
\bot & : \neg [\mathcal{A}] \text{false} \\
T & : [\mathcal{A}] \text{true} \\
\Sigma & : \neg [\emptyset] \neg \varphi \Rightarrow [\Sigma] \varphi \\
M & : [\mathcal{A}] (\varphi_1 \land \varphi_2) \Rightarrow [\mathcal{A}] \varphi_1 \\
S & : [\mathcal{A}] \varphi_1 \lor [\mathcal{A}'] \varphi_2 \Rightarrow [\mathcal{A} \lor \mathcal{A}'] \varphi_1 \lor \varphi_2 , \text{if } \mathcal{A} \land \mathcal{A}' = \emptyset
\end{align*} \]

together with propositional tautologies and the following inference rules: modus ponens (from $\varphi_1$ and $\varphi_1 \Rightarrow \varphi_2$ infer $\varphi_2$) and equivalence (from $\varphi_1 \Leftrightarrow \varphi_2$ infer $[\mathcal{A}] \varphi_1 \Leftrightarrow [\mathcal{A}] \varphi_2$). It can be shown that the inference rule monotonicity (from $\varphi_1 \Rightarrow \varphi_2$ infer $[\mathcal{A}] \varphi_1 \Rightarrow [\mathcal{A}] \varphi_2$) is a derivable rule in this system. The next result will be used later.

Lemma 1 The formula $[\mathcal{A}] \psi_1 \land (\mathcal{B} \setminus \mathcal{A}) \psi_2 \Rightarrow (\mathcal{B} \setminus \mathcal{A}) (\psi_1 \land \psi_2)$ where $\mathcal{A}$ and $\mathcal{B}$ are coalitions, $\mathcal{A} \subseteq \mathcal{B}$, and $\psi_1, \psi_2 \in$ WFF$_{CL}$, is valid.

Proof.
1. $[\mathcal{A}] \psi_1 \land (\mathcal{B} \setminus \mathcal{A}) (\psi_1 \Rightarrow \neg \psi_2) \Rightarrow (\mathcal{B} \setminus \mathcal{A}) (\psi_1 \land (\psi_1 \Rightarrow \neg \psi_2))$ \hspace{1cm} S, $\mathcal{A}' = \mathcal{B} \setminus \mathcal{A}$
2. $\psi_1 \land (\psi_1 \Rightarrow \neg \psi_2) \Rightarrow \neg \psi_2$ \hspace{1cm} propositional tautology
3. $(\mathcal{B} \setminus \mathcal{A}) (\psi_1 \land (\psi_1 \Rightarrow \neg \psi_2)) \Rightarrow (\mathcal{B} \setminus \mathcal{A}) \neg \psi_2$ \hspace{1cm} 2, monotonicity
4. $[\mathcal{A}] \psi_1 \land (\mathcal{B} \setminus \mathcal{A}) (\psi_1 \Rightarrow \neg \psi_2) \Rightarrow (\mathcal{B} \setminus \mathcal{A}) (\neg \psi_1 \lor \neg \psi_2)$ \hspace{1cm} 1,3, chaining
5. $[\mathcal{A}] \psi_1 \land (\mathcal{B} \setminus \mathcal{A}) (\psi_1 \Rightarrow \neg \psi_2) \Rightarrow (\mathcal{B} \setminus \mathcal{A}) (\neg \psi_1 \lor \neg \psi_2)$ \hspace{1cm} 4, rewriting
6. $[\mathcal{A}] \psi_1 \land (\mathcal{B} \setminus \mathcal{A}) (\neg \psi_1 \lor \neg \psi_2) \Rightarrow (\mathcal{B} \setminus \mathcal{A}) (\neg \psi_1 \lor \neg \psi_2)$ \hspace{1cm} 5, def. dual
7. $[\mathcal{A}] \psi_1 \land (\mathcal{B} \setminus \mathcal{A}) \psi_2 \Rightarrow (\mathcal{B} \setminus \mathcal{A}) (\psi_1 \land \psi_2)$ \hspace{1cm} 6, rewriting \hspace{1cm} $\square$

The semantics of CL is given in terms of Concurrent Game Structures (CGS) [2] and it is positional, that is, agents have no memory of their past decisions and, thus, those decisions are made by taking into account only the current state. We note that the semantics of CL is often presented in terms of
**Multiplayer Game Models** (MGMs) \([13]\). Note also that MGMs yield the same set of validities as CGSs \([5]\). As we intend to extend the proof method given here to full ATL, the correctness proofs are based on the tableau procedure for full ATL \([6]\) and we follow the semantics presentation given there.

**Def. 1** A Concurrent Game Frame (CGF) is a tuple \(\mathcal{F} = (\Sigma, \mathcal{I}, s_0, d, \delta)\), where

- \(\Sigma\) is a finite non-empty set of agents;
- \(\mathcal{I}\) is a non-empty set of states, with a distinguished state \(s_0\), termed initial state;
- \(d : \Sigma \times \mathcal{I} \rightarrow \mathbb{N}^+\), where the natural number \(d(a, s) \geq 1\) represents the number of moves that the agent \(a\) has at the state \(s\). Every move for agent \(a\) at the state \(s\) is identified by a number between 0 and \(d(a, s) - 1\). Let \(D(a, s) = \{0, \ldots, d(a, s) - 1\}\) be the set of all moves available to agent \(a\) at \(s\).
- For a state \(s\), a move vector is a \(k\)-tuple \((\sigma_1, \ldots, \sigma_k)\), where \(k = |\Sigma|\), such that \(0 \leq \sigma_a \leq d(a, s) - 1\), for all \(a \in \Sigma\). Intuitively, \(\sigma_a\) represents an arbitrary move of agent \(a\) in \(s\). Let \(D(s) = \Pi_{a \in \Sigma} D(a, s)\) be the set of all move vectors at \(s\). We denote by \(\sigma\) an arbitrary member of \(D(s)\).
- \(\delta\) is a transition function that assigns to every \(s \in \mathcal{I}\) and every \(\sigma \in D(s)\) a state \(\delta(s, \sigma) \in \mathcal{I}\) that results from \(s\) if every agent \(a \in \Sigma\) plays move \(\sigma_a\).

In the following, let \(\mathcal{F} = (\Sigma, \mathcal{I}, s_0, d, \delta)\) be a CGF with \(s, s' \in \mathcal{I}\). We say that \(s'\) is a successor of \(s\) (an \(s\)-successor) if \(s' = \delta(s, \sigma)\), for some \(\sigma \in D(s)\). If \(\kappa\) is a tuple, then \(\kappa_n\) (or \(\kappa(n)\)) denotes the \(n\)-th element of \(\kappa\). Let \(|\Sigma| = k\) and let \(\mathcal{A} \subseteq \Sigma\) be a coalition. An \(\mathcal{A}\)-move \(\sigma_{\mathcal{A}}\) at \(s\) is a \(k\)-tuple such that \(\sigma_{\mathcal{A}}(a) \in D(a, s)\) for every \(a \in \mathcal{A}\) and \(\sigma_{\mathcal{A}}(a') = s\) (i.e. an arbitrary move) for every \(a' \notin \mathcal{A}\). We denote by \(D(\mathcal{A}, s)\) the set of all \(\mathcal{A}\)-moves at state \(s\). A move vector \(\sigma\) extends an \(\mathcal{A}\)-move vector \(\sigma_{\mathcal{A}}\), denoted by \(\sigma_{\mathcal{A}} \subseteq \sigma\) or \(\sigma \supseteq \sigma_{\mathcal{A}}\), if \(\sigma(a) = \sigma_{\mathcal{A}}(a)\) for every \(a \in \mathcal{A}\). Let \(\sigma_{\mathcal{A}} \in D(\mathcal{A}, s)\) be an \(\mathcal{A}\)-move. The outcome of \(\sigma_{\mathcal{A}}\) at \(s\), denoted by \(\text{out}(s, \sigma_{\mathcal{A}})\), is the set of all states \(s' \in \mathcal{I}\) for which there exists a move vector \(\sigma \in D(s)\) such that \(\sigma_{\mathcal{A}} \subseteq \sigma\) and \(\delta(s, \sigma) = s'\).

**Def. 2** A Concurrent Game Model (CGM) is a tuple \(\mathcal{M} = (\mathcal{F}, \Pi, \pi)\), where \(\mathcal{F} = (\Sigma, \mathcal{I}, s_0, d, \delta)\) is a CGF; \(\Pi\) is the set of propositional symbols; and \(\pi : \mathcal{I} \rightarrow 2^\Pi\) is a valuation function.

**Def. 3** Let \(\mathcal{M} = (\Sigma, \mathcal{I}, s_0, d, \delta, \Pi, \pi)\) be a CGM with \(s \in \mathcal{I}\). The satisfaction relation, denoted by \(\models\), is inductively defined as follows.

- \(\langle \mathcal{M}, s \rangle \models \text{true}\);
- \(\langle \mathcal{M}, s \rangle \models p\) iff \(p \in \pi(s)\), for all \(p \in \Pi\);
- \(\langle \mathcal{M}, s \rangle \models \neg \phi\) iff \(\langle \mathcal{M}, s \rangle \not\models \phi\);
- \(\langle \mathcal{M}, s \rangle \models \phi \land \psi\) iff \(\langle \mathcal{M}, s \rangle \models \phi\) and \(\langle \mathcal{M}, s \rangle \models \psi\);
- \(\langle \mathcal{M}, s \rangle \models \langle \mathcal{A}\rangle \phi\) iff there exists a \(\mathcal{A}\)-move \(\sigma_{\mathcal{A}} \in D(\mathcal{A}, s)\) s.t. \(\langle \mathcal{M}, s' \rangle \models \phi\) for all \(s' \in \text{out}(s, \sigma_{\mathcal{A}})\);
- \(\langle \mathcal{M}, s \rangle \models \langle \mathcal{A}\rangle \phi\) iff for all \(\mathcal{A}\)-moves \(\sigma_{\mathcal{A}} \in D(\mathcal{A}, s)\) exists \(s' \in \text{out}(s, \sigma_{\mathcal{A}})\) s.t. \(\langle \mathcal{M}, s' \rangle \models \phi\).

Semantics of false, disjunctions, and implications are given in the usual way. Given a model \(\mathcal{M}\), a state \(s\) in \(\mathcal{M}\), and a formula \(\varphi\), if \(\langle \mathcal{M}, s \rangle \models \varphi\), we say that \(\varphi\) is satisfied at the state \(s\) in \(\mathcal{M}\).

In this work, we consider tight satisfiability, i.e. the evaluation of a formula \(\varphi\) depends only on the agents occurring in \(\varphi\) \([16]\). We denote by \(\Sigma_\varphi\), where \(\Sigma_\varphi \subseteq \Sigma\), the set of agents occurring in a well-formed formula \(\varphi\). If \(\Phi\) is a set of well-formed formulae, \(\Sigma_\Phi \subseteq \Sigma\) denotes \(\bigcup_{\varphi \in \Phi} \Sigma_\varphi\). Let \(\varphi \in \text{WFF}_\text{CL}\) and \(\mathcal{M} = (\Sigma_\varphi, \mathcal{I}, s_0, d, \delta, \Pi, \pi)\) be a CGM. Formulae are interpreted with respect to the distinguished world \(s_0\). Thus, a formula \(\varphi\) is said to be satisfiable in \(\mathcal{M}\), denoted by \(\mathcal{M} \models \varphi\), if \(\langle \mathcal{M}, s_0 \rangle \models \varphi\); it is said to be satisfiable if there is a model \(\mathcal{M}\) such that \(\langle \mathcal{M}, s_0 \rangle \models \varphi\); and it is said to be valid if for all models \(\mathcal{M}\) we have \(\langle \mathcal{M}, s_0 \rangle \models \varphi\). A finite set \(\Gamma \subseteq \text{WFF}_\text{CL}\) is satisfiable in a model \(\mathcal{M}, s\) in \(\mathcal{M}\), denoted by \(\langle \mathcal{M}, s \rangle \models \Gamma\), if for all \(\varphi \in \Gamma, 0 \leq i \leq n, n \in \mathbb{N}\), \(\langle \mathcal{M}, s \rangle \models \varphi_i\); \(\Gamma\) is satisfiable in a model \(\mathcal{M}\), \(\mathcal{M} \models \Gamma\), if \(\langle \mathcal{M}, s_0 \rangle \models \Gamma\); and \(\Gamma\) is satisfiable, if there is a model \(\mathcal{M}\) such that \(\mathcal{M} \models \Gamma\).
3 Resolution Calculus

The resolution calculus for CL, $\text{RES}_{\text{CL}}$, operates on sets of clauses. A formula in CL is firstly converted into a coalition problem, which is then transformed into a coalition problem in Divided Separated Normal Form for Coalition Logic, $\text{DSNF}_{\text{CL}}$.

**Def. 4** A coalition problem is a tuple $(\mathcal{I}, \mathcal{U}, \mathcal{N})$, where $\mathcal{I}$, the set of initial formulae, is a finite set of propositional formulae; $\mathcal{U}$, the set of global formulae, is a finite set of formulae in $\text{WFF}_{\text{CL}}$; and $\mathcal{N}$, the set of coalition formulae, is a finite set of coalition formulae, i.e. those formulae in which a coalition modality occurs.

The semantics of coalition problems assumes that initial formulae hold at the initial state; and that global and coalition formulae hold at every state of a model.

**Def. 5** Given a coalition problem $\mathcal{C} = (\mathcal{I}, \mathcal{U}, \mathcal{N})$, we denote by $\Sigma_{\mathcal{C}}$ the set of agents $\Sigma_{\mathcal{U}\cup\mathcal{N}}$. If $\mathcal{C} = (\mathcal{I}, \mathcal{U}, \mathcal{N})$ is a coalition problem and $\mathcal{M} = (\Sigma_{\mathcal{C}}, \mathcal{I}, s_0, d, \pi)$ is a CGM, then $\mathcal{M} \models \mathcal{C}$ if, and only if, $(\mathcal{M}, s_0) \models \mathcal{I}$ and $(\mathcal{M}, s) \models \mathcal{U} \cup \mathcal{N}$, for all $s \in \mathcal{I}$. We say that $\mathcal{C} = (\mathcal{I}, \mathcal{U}, \mathcal{N})$ is satisfiable, if there is a model $\mathcal{M}$ such that $\mathcal{M} \models \mathcal{C}$.

In order to apply the resolution method, we further require that formulae within each of those sets are in clausal form: initial clauses and global clauses are of the form $\bigvee_{j=1}^n l_j$; positive coalition clauses are of the form $\bigwedge_{i=1}^m l_i \Rightarrow (\mathcal{A}) \bigvee_{j=1}^n l_j$; and negative coalition clauses are of the form $\bigwedge_{i=1}^m l_i \Rightarrow (\mathcal{A}) \bigwedge_{j=1}^n l_j$; where $m, n \geq 0$ and $l_i, l_j$, for all $1 \leq i \leq m$, $1 \leq j \leq n$, are literals or constants. We assume that clauses are kept in the simplest form by means of usual Boolean simplification rules. Tautologies are removed from the set of clauses as they cannot contribute to finding a contradiction. A coalition problem in $\text{DSNF}_{\text{CL}}$ is a coalition problem $(\mathcal{I}, \mathcal{U}, \mathcal{N})$ such that $\mathcal{I}$ is a set of initial clauses, $\mathcal{U}$ is a set of global clauses, and $\mathcal{N}$ is a set of positive and negative coalition clauses.

The transformation of a coalition logic formula into a coalition problem in $\text{DSNF}_{\text{CL}}$ is analogous to the approach taken in [4]. The transformation of a formula into a coalition problem in $\text{DSNF}_{\text{CL}}$, which is given in [11, 12], reduces the number of operators and separates the contexts to which the resolution inference rules are applied, but may add new propositional symbols.

The set of inference rules for $\text{RES}_{\text{CL}}$ are given as follows. Let $(\mathcal{I}, \mathcal{U}, \mathcal{N})$ be a coalition problem in $\text{DSNF}_{\text{CL}}$: $C, C'$ be conjunctions of literals; $D, D'$ be disjunctions of literals; $l, l'$ be literals; and $\mathcal{A}, \mathcal{B} \subseteq \Sigma$ be coalitions (where $\Sigma$ is the set of all agents). The first rule, $\text{GRES1}$, is classical resolution applied to clauses which are true at the initial state. The next inference rule, $\text{GRES1}$, performs resolution on clauses which are true in all states.

$$\text{GRES1} \quad D \lor l \quad \in \mathcal{I} \quad \text{GRES1} \quad D \lor l \quad \in \mathcal{U}$$
$$D' \lor \neg l \quad \in \mathcal{I} \cup \mathcal{U} \quad D' \lor \neg l \quad \in \mathcal{U}$$
$$D \lor D' \quad \in \mathcal{I} \quad D \lor D' \quad \in \mathcal{U}$$

Soundness of $\text{GRES1}$ and $\text{GRES1}$ follow from the semantics of coalition problems and the soundness result for classical propositional resolution [15]. The following rules perform resolution on positive and negative coalition clauses.

$$\text{GRES2} \quad D \lor l \quad \in \mathcal{U}$$
$$D' \lor \neg l \quad \in \mathcal{U}$$
$$C \quad \Rightarrow \quad [\mathcal{A}] (D \lor l) \quad \in \mathcal{N}$$
$$C \quad \Rightarrow \quad [\mathcal{A}] (D' \lor \neg l) \quad \in \mathcal{N}$$
$$C' \quad \Rightarrow \quad [\mathcal{B}] (D' \lor \neg l) \quad \in \mathcal{N}$$
$$C' \quad \Rightarrow \quad (\mathcal{A} \lor \mathcal{B}) (D \lor D') \quad \in \mathcal{N}$$
$$C \quad \Rightarrow \quad \langle \mathcal{A} \rangle (D \lor l) \quad \in \mathcal{N}$$
$$C \quad \Rightarrow \quad \langle \mathcal{A} \rangle (D' \lor l) \quad \in \mathcal{N}$$
$$C \quad \Rightarrow \quad \langle \mathcal{A} \rangle (D \lor l) \quad \in \mathcal{N}$$
$$C \quad \Rightarrow \quad \langle \mathcal{A} \rangle (D' \lor \neg l) \quad \in \mathcal{N}$$
Soundness of the inference rules CRES1-4 follow from the axiomatisation of CL, given in Section 2. We give sketches of the proofs here. Let $\mathcal{M}$ be a CGM and $s \in \mathcal{M}$ a state. Recall that coalition clauses are satisfied at any state in $\mathcal{M}$. For CRES1, if $\langle \mathcal{M}, s \rangle \models C \land C'$, by the semantics of conjunction and implication, we have that $\langle \mathcal{M}, s \rangle \models C \land C' \Rightarrow [\mathcal{A}](D \lor l) \land [\mathcal{B}](D \lor 0)$. By axiom $S$, we have that $[\mathcal{A}](D \lor l) \land [\mathcal{B}](D \lor 0)$ implies $[\mathcal{A} \cup \mathcal{B}](D \lor l) \land (D \lor 0)$). Therefore, $\langle \mathcal{M}, s \rangle \models C \land C' \Rightarrow [\mathcal{A} \cup \mathcal{B}](D \lor l) \land (D \lor 0))$. By classical resolution applied within the successor states, we obtain that $\langle \mathcal{M}, s \rangle \models C \land C' \Rightarrow [\mathcal{A} \cup \mathcal{B}](D \lor l) \land (D \lor 0))$. For CRES3, by Lemma 1, we have that $[\mathcal{A}](D \lor l) \land [\mathcal{B}](D \lor 0)$ implies $[\mathcal{A} \cup \mathcal{B}](D \lor l) \land (D \lor 0)$), with $A \subseteq B$, is valid. If $\langle \mathcal{M}, s \rangle \models C \land C'$, by the semantics of implication, we have that $\langle \mathcal{M}, s \rangle \models C \land C' \Rightarrow [\mathcal{A} \cup \mathcal{B}](D \lor l) \land (D \lor 0)$). Applying classical resolution within the successor states, we obtain that $\langle \mathcal{M}, s \rangle \models C \land C' \Rightarrow [\mathcal{A} \cup \mathcal{B}](D \lor l) \land (D \lor 0))$. Soundness of the inference rules CRES2 and CRES4 follow from the above and the semantics of coalition problems: as $D \lor l$ in $\mathcal{U}$ is satisfied at all states, we have that true $\Rightarrow [\mathcal{A}](D \lor l)$ is also satisfied at all states.

The next two inference rules are justified by the axioms $\bot$ and $\top$, given by $[\mathcal{A}]\text{false}$ and $[\mathcal{A}]\text{true}$, respectively, which imply that the consequent in both rewriting rules cannot be satisfied.

\[
\begin{align*}
\text{RW1} & \quad \frac{\bigwedge_{i=1}^{n} l_i \Rightarrow [\mathcal{A}]\text{false}}{\bigvee_{i=1}^{n} \neg l_i} \in \mathcal{N} \\
\text{RW2} & \quad \frac{\bigwedge_{i=1}^{n} l_i \Rightarrow [\mathcal{A}]\text{false}}{\bigvee_{i=1}^{n} \neg l_i} \in \mathcal{U} 
\end{align*}
\]

As sketched above, the resolution-based calculus for Coalition Logic is sound.

**Theorem 1 (Soundness)** Let $\mathcal{C}$ be a coalition problem in DSF CL. Let $\mathcal{C}'$ be the coalition problem in DSF CL obtained from $\mathcal{C}$ by applying any of the inference rules IRES1, GRES1, CRES1-4 or RW1-2 to $\mathcal{C}$. If $\mathcal{C}$ is satisfiable, then $\mathcal{C}'$ is satisfiable.

A derivation from a coalition problem in DSF CL $\mathcal{C} = (\mathcal{I}, \mathcal{U}, \mathcal{N})$ by RESCL is a sequence $\mathcal{C}_0, \mathcal{C}_1, \mathcal{C}_2, \ldots$ of problems such that $\mathcal{C}_0 = \mathcal{C}$, $\mathcal{C}_i = (\mathcal{I}, \mathcal{U}_i, \mathcal{N}_i)$, and $\mathcal{C}_{i+1}$ is either $(\mathcal{I} \cup \{ D \}, \mathcal{U}_i \cup \mathcal{N}_i)$, where $D$ is the conclusion of IRES1; $(\mathcal{I}, \mathcal{U}_i \cup \{ D \}, \mathcal{N}_i)$, where $D$ is the conclusion of GRES1, RW1, or RW2; or $(\mathcal{I}, \mathcal{U}_i, \mathcal{N}_i \cup \{ D \})$, where $D$ is the conclusion of CRES1, CRES2, CRES3, or CRES4; and $D$ is not a tautology.

A refutation for a coalition problem in DSF CL $\mathcal{C} = (\mathcal{I}, \mathcal{U}, \mathcal{N})$ (by RESCL) is a derivation from $\mathcal{C}$ such that for some $i \geq 0$, $\mathcal{C}_i = (\mathcal{I}, \mathcal{U}_i, \mathcal{N}_i)$ contains a contradiction, where a contradiction is given by either false $\in \mathcal{I}$ or false $\in \mathcal{U}_i$. A derivation terminates if, and only if, either a contradiction is derived or no new clauses can be derived by further application of resolution rules of RESCL.

The completeness proof for RESCL is based on the tableau construction given in [6]. Given an unsatisfiable coalition problem in DSF CL $\mathcal{C}$, an initial tableau is obtained by this construction which is then reduced to an empty tableau via a sequence of deletion steps. We show that each deletion step corresponds to an application of the resolution inference rules to (sub)sets of clauses in $\mathcal{C}$ or clauses previously derived from $\mathcal{C}$. The derivation constructed in this way is shown to be a refutation of $\mathcal{C}$.

**Theorem 2 (Completeness)** Let $\mathcal{C} = (\mathcal{I}, \mathcal{U}, \mathcal{N})$ be an unsatisfiable coalition problem in DSF CL. Then there is a refutation for $\mathcal{C}$ using the inference rules IRES1, GRES1, CRES1-4, and RW1-2.

The proof that every derivation terminates is trivial and based on the fact that we have a finite number of clauses that can be expressed. As the number of propositional symbols after translation into the normal form is finite and the inference rules do not introduce new propositional symbols, we have that the number of possible literals occurring in clauses is finite and the number of conjunctions (resp. disjunctions) on the left-hand side (resp. right-hand side) of clauses is finite (modulo simplification). As the number of agents is finite, the number of coalition modalities that can be introduced by inference rules is also finite. Thus, only a finite number of clauses can be expressed (modulo simplification), so at some point either we derive a contradiction or no new clauses can be generated.
Theorem 3 Let $\mathcal{C} = (\mathcal{I}, \mathcal{U}, \mathcal{N})$ be a coalition problem in DSNF$_{CL}$. Then any derivation of $\mathcal{C}$ by RES$_{CL}$ terminates.

Full proofs for soundness, completeness, termination, and complexity of the resolution-based method for CL are given in [11][12].

Example 1 We show a simple example, adapted from [9], of the application of RES$_{CL}$ to a problem involving the cooperation of agents. There are two agents (1 and 2) and two toggle switches. For each agent $a = 1, 2$, there are two possible actions: $[a]\text{tog}_a \land [a]\neg\text{tog}_a$, where $\text{tog}_a$ denotes that the agent $a$ can toggle the switch $a$ (clauses 3, 9–13). The light is initially off, i.e. we have that $t_0 \Rightarrow \neg l$ (clauses 1 and 2). If the light is off and the switch is toggled, then at the next moment the light is on: $\text{tog}_a \land \neg l \Rightarrow [a]l$ (clauses 5 and 6). Similarly, if the light is on and the agent toggles the switch, then at the next moment the light is off: $\text{tog}_a \land l \Rightarrow [a]\neg l$ (clauses 7 and 8). We prove that the agents can cooperate to turn on the light, that is, we introduce the clauses 4 and 14, which corresponds to the negation of $[1, 2]l$.

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>$t_0$</td>
<td>$[\mathcal{I}]$</td>
</tr>
<tr>
<td>2.</td>
<td>$\neg t_0 \lor \neg l$</td>
<td>$[\mathcal{U}]$</td>
</tr>
<tr>
<td>3.</td>
<td>$\neg t_0 \lor t_1$</td>
<td>$[\mathcal{U}]$</td>
</tr>
<tr>
<td>4.</td>
<td>$\neg t_1 \lor t_4$</td>
<td>$[\mathcal{U}]$</td>
</tr>
<tr>
<td>5.</td>
<td>$\text{tog}_1 \land \neg l$</td>
<td>$[\mathcal{N}]$</td>
</tr>
<tr>
<td>6.</td>
<td>$\text{tog}_2 \land \neg l$</td>
<td>$[\mathcal{N}]$</td>
</tr>
<tr>
<td>7.</td>
<td>$\text{tog}_1 \land l$</td>
<td>$[\mathcal{N}]$</td>
</tr>
<tr>
<td>8.</td>
<td>$\text{tog}_2 \land l$</td>
<td>$[\mathcal{N}]$</td>
</tr>
<tr>
<td>9.</td>
<td>$t_1 \Rightarrow [a]t_1 [\mathcal{A}]$</td>
<td></td>
</tr>
<tr>
<td>10.</td>
<td>$t_1 \Rightarrow [a]t_2 [\mathcal{A}]$</td>
<td></td>
</tr>
<tr>
<td>11.</td>
<td>$t_1 \Rightarrow [a]\neg t_1 [\mathcal{A}]$</td>
<td></td>
</tr>
<tr>
<td>12.</td>
<td>$t_1 \Rightarrow [a]\neg t_2 [\mathcal{A}]$</td>
<td></td>
</tr>
<tr>
<td>13.</td>
<td>$[\emptyset]t_1 [\mathcal{N}]$</td>
<td></td>
</tr>
</tbody>
</table>

4 CLProver

CLProver is a prototype implementation of the resolution-based method given in [12]. The prover is written in SWI-Prolog (Multi-threaded, 64 bits, Version 6.0.2) and the compiled binaries for Linux x86_64 together with instructions for usage and example files are available at [http://www.cic.unb.br/docentes/nalon/#software](http://www.cic.unb.br/docentes/nalon/#software).

The prover recurs over the set of clauses using breadth-first search for a proof. The resolution inference rules for CL are in fact variations of the propositional resolution rule. Before presenting the general form of the inference rules, we explain the data structures that are employed by the prover. A clause core is implemented as a list with three elements, all of which are lists: a list of literals on the left-hand side of a clause, a list of agents, and a list of literals on the right-hand side of a clause. The only operator allowed within the lists of literals is the negation operator, neg. Clauses are then given as Prolog lists, with four elements. The first element is the clause number, the second is the clause core, the third is the justification (‘given’, if the clause is an input clause; or a list containing the numbers of the clauses from which it was derived, together with the literal being resolved, and the inference rule applied), and the fourth is an indication to which set within a coalition problem the clause belongs (‘i’ for initial, ‘u’ for global, ‘p’ for positive, and ‘n’ for negative). Thus, for instance, the clauses 1, 3, and 20 from Example[1]...
are represented as \([1,\\emptyset,\emptyset,[t0]], \text{[given]}, i\), \([3,\\emptyset,\emptyset,[\neg t0, t1]], \text{[given]}, u\), and \([20,[[t1],[1],[1]], [9,19,\text{tog1},\text{cres1}], n]\), respectively.

Given this representation, the propositional resolution inference rule is modified in such a way that a clause \([_,[LHS1],[AG1],[RHS1],_,S1]\) is resolved with \([_,[LHS2],[AG2],[RHS2],_,S2]\), if such clauses meet the side conditions given by the inference rules presented in Section 3. For instance, the rule \textsc{CRES1} is applied if both \(S1\) and \(S2\) are equal to ‘p’ and if the intersection between \(LHS1\) and \(LHS2\) is empty. The prover then recurs over the set of initial, global and coalition clauses using the following procedure (where \(S\) is a saturated set of clauses and \(N\) is a non-saturated set of clauses):

\[
\text{procedure } \text{resolution}(S, N) \\
\text{while } (N \neq \emptyset \text{ and false } \not\in N) \\
\text{do } \text{Given } \leftarrow \text{choose}(N); \\
N \leftarrow N \setminus \{\text{Given}\}; \\
S \leftarrow S \cup \{\text{Given}\}; \\
\text{New } \leftarrow \text{rewrite}(\text{res}(\text{Given},S)); \\
\text{/* Forward Subsumption */} \\
N \leftarrow \text{sub}(\text{sub}(\text{New},S),N); \\
\text{end-while} \\
\text{if } \text{false } \in N \text{ then } S \leftarrow S \cup \{\text{false}\}; \\
\text{return } S;
\]

where \(\text{choose}(N)\) randomly picks a clause in \(N\); \(\text{res}(C, N)\) is the set of all non-tautological resolvents in simplified form derivable between a clause \(C\) and a set of clauses \(N\) by one of the inference rules; \(\text{rewrite}(N)\) is the union of \(N\) and the set of clauses derived by the rewriting rules; and \(\text{sub}(M,N)\) is the set of clauses in \(M\) not subsumed by a clause in \(N\). Forward subsumption is implemented for both the propositional and modal portions of the language. For the propositional part, a clause \(D\) in \(I\) (resp. \(U\)) is subsumed by a clause \(D'\) in \(I \cup \mathcal{U}\) (resp. \(\mathcal{U}\)) if \(D' \subseteq D\). A positive coalition clause \(C \Rightarrow [\alpha]D\) is subsumed by another positive coalition clause \(C' \Rightarrow [\alpha']D'\), if \(C' \subseteq C\), \(\alpha' \subseteq \alpha\), and \(D' \subseteq D\). A negative coalition clause \(C \Rightarrow \langle \alpha \rangle D\) is subsumed by another negative coalition clause \(C' \Rightarrow \langle \alpha' \rangle D'\), if \(C' \subseteq C\), \(\alpha \subseteq \alpha'\), and \(D' \subseteq D\). Some other forms of subsumption have not been implemented in the current version of the prover, as, for instance, coalition clauses which are subsumed by global clauses.

The current version of \textsc{CLProver} is a prototype. The prover implements subsumption, but it does not implement any of the usual performance improving techniques for resolution-based methods. For example, the function \(\text{choose}(N)\) does not use any heuristic to determine what given clause to pick. Further refinements of the resolution calculus, for example, ordered resolution or the the use of a set of support strategy would also greatly improve the performance of the prover. \textsc{CLProver}, however, performs well when compared with both versions of another tool, namely, \textsc{TATL}, a tableau-based prover for ATL \cite{3}.

In the following, \textsc{TATL-A} refers to the April version of the \textsc{TATL} prover, available at \url{https://www.ibisc.univ-evry.fr/~adavid/bin/tatl.tar.gz}; and \textsc{TATL-N} refers to the November version, available at \url{http://atila.ibisc.univ-evry.fr/tableau_ATL/bin/tatl.tar.gz}.

A benchmark, consisting of six sets of randomly generated \(\text{CL}\) formulae, was designed to compare the performance of both provers. The formulae in the benchmark are characterised by five parameters: (1) the number of propositional symbols \(N\); (2) the number of agents \(A\); (3) the number of conjuncts \(L\); (4) the modal degree \(D\); and (5) the probability \(P\). Based on a given choice of parameters random formulae in conjunctive normal form (CNF) are defined inductively as follows. A \textit{random (coalition) atom} of degree 0 is a propositional variable randomly chosen from the set of \(N\) propositional symbols. A \textit{random coalition atom} of degree \(D, D > 0\), is with probability \(P\): (a) an expression of the form \([\alpha] \varphi\),
where $[\mathcal{A}]$ is a coalition modality with a set of agents $\mathcal{A}$ randomly chosen from $\mathcal{A}^{\{1,\ldots,A\}}$ and $\varphi$ is a random coalition CNF clause of modal degree $D-1$ (defined below), or (b) a random atom of degree 0, otherwise. A random coalition literal (of degree $D$) is with probability 0.5 a random coalition atom (of degree $D$) or its negation, otherwise. A random coalition CNF clause (of degree $D$) is a disjunction of three random coalition literals (of degree $D$). A random coalition CNF formula (of degree $D$) is a conjunction of $L$ random coalition CNF clauses (of degree $D$).

The six sets of problems used to compare CLProver and TATL were generated using $N=5$, $A=2$, $L \in \{5,\ldots,10\}$, $D=1$, and $P=1$. The experiment was run on an i7-3537U CPU at 3.00GHz, 8GB RAM, under Linux kernel 3.11.10-100. The average run-time for each set of problems is shown in Figure 1. The provers were given a timeout of 1000 seconds. CLProver has solved all problems in all sets within the given time. TATL-A has failed to solve any problems in the sets 5-2-009-1 and 5-2-010-1. TATL-N has solved all problems in the sets 5-2-005-1 and 5-2-006-1; nine problems in 5-2-007-1; three in 5-2-008-1; four in 5-2-009-1; and none in 5-2-010-1. For the calculation of the average times, whenever the prover has timed out, we have set the corresponding time to 1000 seconds.

5 Conclusion

The resolution-based calculus for the Coalition Logic CL is applied to a coalition problem in DSNF$_{CL}$, which separates the dimensions to which the resolution rules are applied. The transformation into the normal form is satisfiability preserving and polynomially bounded by the size of the original formula. Soundness of the method follows from the axiomatisation of CL. Completeness is proved with respect to the tableau procedure given in [6]: if a tableau for a coalition problem is closed, there is a refutation based on the calculus given here. Termination is ensured by the fact that number of propositional symbols and agents is finite, so there are only a finite number of clauses that can be generated.

The decision procedure based on RES$_{CL}$ is in EXPTIME, as shown in [12]. This is optimal, as the satisfiability problem for coalition problems in DSNF$_{CL}$ is EXPTIME-hard, thus more expressive than...
the language of CL. This result follows from [16, Lemma 4.10, page 785] and the fact that an extension of CL with positive occurrences of ATL’s $\langle(0)\rangle$ operator can be translated into DSNF$_{\text{CL}}$. It also follows that DSNF$_{\text{CL}}$ is more expressive than CL.

**CLProver** is the first (prototype) implementation of RES$_{\text{CL}}$. The experiments we have performed suggest that it is a viable tool for reasoning about Coalition Logic. Future work includes further improvements to CLProver. We also intend to extend our calculus to the full language of ATL.

**References**


