Ordered Resolution for Coalition Logic

Ullrich Hustadt[1], Paul Gainer[1], Clare Dixon[1], Cláudia Nalon[2], and Lan Zhang[3]⋆

[1] Department of Computer Science, University of Liverpool, UK
{uhustadt,sgpgaine,cldixon}@liverpool.ac.uk
[2] Department of Computer Science, University of Brasília, Brazil
nalon@unb.br
[3] Information School, Capital University of Economics and Business, China
lan@cueb.edu.cn

Abstract In this paper we introduce a calculus based on ordered resolution for Coalition Logic (CL), improving our previous approach based on unrefined resolution, and discuss the problems associated with imposing an ordering refinement in the context of CL. The calculus operates on 'coalition problems', a normal form for CL where we use coalition vectors that can represent choices made by agents explicitly, and the inference rules of the calculus provide the basis for a decision procedure for the satisfiability problem in CL. We give correctness, termination and complexity results for our calculus. We also present experimental results for an implementation of the calculus and show that it outperforms a tableau-based decision procedure for Alternating-Time Temporal Logic (ATL) on two classes of benchmark formulae for CL.

1 Introduction

Coalition Logic CL was introduced by Pauly [18] as a logic for reasoning about what groups of agents can bring about by collective action. CL is a multi-modal logic with modal operators of the form $[\mathcal{A}]$, where $\mathcal{A}$ is a set of agents. The formula $[\mathcal{A}]\varphi$, where $\mathcal{A}$ is a set of agents and φ is a formula, can be read as *the coalition of agents $\mathcal{A}$ can bring about φ* or *the coalition of agents $\mathcal{A}$ is effective for φ* or *the coalition of agents $\mathcal{A}$ has a strategy to achieve φ*. Applications of Coalition Logic include the verification of properties of voting procedures and reasoning about strategic games [18].

Coalition Logic is closely related to *Alternating-Time Temporal Logic* ATL [1], a multi-modal logic with coalition quantifiers $\langle\langle\mathcal{A}\rangle\rangle$, where $\mathcal{A}$ is again a set of agents, and temporal operators $\bigcirc$ ("next"), $\Box$ ("always") and $\mathcal{U}$ ("until"), that extends propositional logic with formulae of the form $\langle\langle\mathcal{A}\rangle\rangle\bigcirc\varphi$, $\langle\langle\mathcal{A}\rangle\rangle\Box\varphi$ and $\langle\langle\mathcal{A}\rangle\rangle\varphi\,\mathcal{U}\,\psi$. CL is equivalent to the next-time fragment of ATL [9], where $[\mathcal{A}]\varphi$ translates into $\langle\langle\mathcal{A}\rangle\rangle\bigcirc\varphi$ (read as *the coalition $\mathcal{A}$ can ensure φ at the next moment*

⋆ The second author was supported by a UK EPSRC Vacation Bursary. The fifth author is supported by the National Natural Science Foundation of China (Grant No. 61303018).

in time). The satisfiability problems for ATL and CL are EXPTIME-complete [23] and PSPACE-complete [18], respectively.

Methods for tackling the satisfiability problem for these logics include two tableau-based methods for ATL [23,11], two automata-based methods [7,10] for ATL, and one tableau-based method for CL [12]. An implementation of the two-phase tableau calculus by Goranko and Shkatov for ATL [11] exists in the form of TATL [6]. A first resolution-based method for CL, $\mathsf{RES_{CL}}$, consisting of a normal form transformation and a resolution calculus, was presented in [16] and shown to be sound, complete and terminating. In particular, the completeness of $\mathsf{RES_{CL}}$ is shown relative to the tableau calculus for ATL in [11]. If a CL formula φ is unsatisfiable, the corresponding tableau is closed. In the completeness proof for $\mathsf{RES_{CL}}$ it is shown that deletions that produce the closed tableau correspond to applications of the resolution inference rules of $\mathsf{RES_{CL}}$ that in turn produce a refutation of φ. A prototype implementation of $\mathsf{RES_{CL}}$ in the programming language Prolog exists in the form of $\mathsf{CLProver}$ [17].

In this paper we revisit the calculus $\mathsf{RES_{CL}}$ for CL and its implementation. $\mathsf{RES_{CL}}$ is based on unrefined propositional resolution. It is natural from a theoretical perspective, as well as vital for practical applications, to consider the question whether refinements of propositional resolution carry over to $\mathsf{RES_{CL}}$. In this paper we focus on an ordering refinement, one of the most commonly used refinements of resolution for both non-classical logics [13,14,25] and classical logic [4]. First, we discuss why the naive use of an ordering to restrict inferences in $\mathsf{RES_{CL}}$ leads to incompleteness. Second, we introduce a new normal form for CL that via so-called coalition vectors represent choices made by agents explicitly. This new normal form allows us to devise the calculus $\mathsf{RES^{\succ}_{CL}}$, a sound and complete ordered resolution calculus for CL. Finally, we provide an experimental evaluation and comparison of $\mathsf{CLProver}$++, an implementation of $\mathsf{RES^{\succ}_{CL}}$ in C++, with $\mathsf{CLProver}$ and TATL.

The paper is organised as follows. In the next section, we present the syntax and semantics of CL. In Section 3, we introduce the normal form transformation for CL and the resolution calculus $\mathsf{RES^{\succ}_{CL}}$. Section 4 motivates our approach to ordered resolution for CL, defines the new normal form for CL, and describes the calculus $\mathsf{RES^{\succ}_{CL}}$. Section 5 briefly describes $\mathsf{CLProver}$++ and presents the evaluation and comparison with other theorem provers for CL. Conclusions and future work are given in Section 6.

2 Coalition Logic

Let $\Sigma \subset \mathbb{N}$ be a non-empty, finite set of agents and $\Pi = \{p, q, \ldots, p_1, q_1, \ldots\}$ be a non-empty, finite or countably infinite set of propositional symbols. A *coalition* $\mathcal{A}$ is a finite subset of Σ. Formulae in CL are constructed from propositional symbols using Boolean operators and the *coalition modalities* $[\mathcal{A}]$ and $\langle\mathcal{A}\rangle$.

Definition 1. *The set* $\mathit{WFF}_{\mathsf{CL}}$ *of* CL formulae *is inductively defined as follows.*
– *all propositional symbols in* Π *are* CL *formulae;*

– *if φ and ψ are* CL *formulae, then so are $\neg\varphi$ (negation) and $(\varphi \to \psi)$ (implication);*
– *if φ_i, $1 \leq i \leq n$, $n \in \mathbb{N}_0$, are* CL *formula, then so are $(\varphi_1 \wedge \ldots \wedge \varphi_n)$ (conjunction), also written $\bigwedge_{i=1}^{n} \varphi_i$, and $(\varphi_1 \vee \ldots \vee \varphi_n)$ (disjunction), also written $\bigvee_{i=1}^{n} \varphi_i$; and*
– *if $\mathcal{A} \subseteq \Sigma$ is a finite set of agents and φ is a* CL *formula, then so are $[\mathcal{A}]\varphi$ (*positive coalition formula*) and $\langle\mathcal{A}\rangle\varphi$ (*negative coalition formula*).*

Parentheses will be omitted if the reading is not ambiguous. We consider the conjunction and disjunction operators to be associative and commutative, that is, we do not distinguish between, for example, $(p \vee (q \vee r))$, $((r \vee p) \vee q)$ and $(q \vee r \vee p)$. The formula $\bigvee_{i=1}^{0} \varphi_i$ is called the *empty disjunction*, also denoted by **false**, while $\bigwedge_{i=1}^{0} \varphi_i$ is called the *empty conjunction*, also denoted by **true**. When enumerating a specific set of agents, we often omit the curly brackets. For example, we write $[1,2]\varphi$ instead of $[\{1,2\}]\varphi$, for a formula φ. A *coalition formula* is either a positive or a negative coalition formula. In the following, we use "formula(e)" and "well-formed formula(e)" interchangeably.

Definition 2. *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 use *Concurrent Game Structures* (CGSs) [1,11] for describing the semantics of ATL. Also, the semantics given here uses *rooted models*, that is, models with a distinguished state where a formula has to be satisfied. Concurrent Game Structures yield the same set of validities as *Multiplayer Game Models* (MGMs) [9] that were originally used by Pauly [18] to define the semantics of Coalition Logic.

Definition 3. *A* Concurrent Game Frame *(CGF) over Σ with root s_0 is a tuple $\mathcal{F} = (\Sigma, \mathcal{S}, s_0, d, \delta)$, where*

– *Σ is a finite non-empty set of* agents*;*
– *$\mathcal{S}$ is a non-empty set of* states*, with a distinguished state s_0;*
– *$d : \Sigma \times \mathcal{S} \longrightarrow \mathbb{N}_0^+$, 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 tuple σ we use $\sigma(n)$ to refer to the n-th element of σ. For a state s, a* move vector *σ 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 a 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 σ an arbitrary member of $D(s)$.*
– *δ is a* transition function *that assigns to every $s \in \mathcal{S}$ and every $\sigma \in D(s)$ a state $\delta(s,\sigma) \in \mathcal{S}$ that results from s if every agent $a \in \Sigma$ plays move $\sigma(a)$.*

Given a CGF $\mathcal{F} = (\Sigma, \mathcal{S}, s_0, d, \delta)$ with $s, s' \in \mathcal{S}$, we say that s' is a *successor* of s (an s-successor) if $s' = \delta(s,\sigma)$, for some $\sigma \in D(s)$.

Definition 4. *Let $|\Sigma| = k$ and let $\mathcal{A} \subseteq \Sigma$ be a coalition. An $\mathcal{A}$-*move *$\sigma_{\mathcal{A}}$ at $s \in \mathcal{S}$ is a k-tuple such that $\sigma_{\mathcal{A}}(a) \in D(a, s)$ for every $a \in \mathcal{A}$ and $\sigma_{\mathcal{A}}(a') = *$ (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.*

Definition 5. *A move vector σ* extends *an $\mathcal{A}$-move $\sigma_{\mathcal{A}}$, denoted by $\sigma_{\mathcal{A}} \sqsubseteq \sigma$ or $\sigma \sqsupseteq \sigma_{\mathcal{A}}$, if $\sigma(a) = \sigma_{\mathcal{A}}(a)$ for every $a \in \mathcal{A}$.*

Given a coalition $\mathcal{A} \subseteq \Sigma$, an $\mathcal{A}$-move $\sigma_{\mathcal{A}} \in D(\mathcal{A}, s)$, and a $\Sigma \setminus \mathcal{A}$-move $\sigma_{\Sigma \setminus \mathcal{A}} \in D(\Sigma \setminus \mathcal{A}, s)$, we denote by $\sigma_{\mathcal{A}} \sqcup \sigma_{\Sigma \setminus \mathcal{A}}$ the unique $\sigma \in D(s)$ such that both $\sigma_{\mathcal{A}} \sqsubseteq \sigma$ and $\sigma_{\Sigma \setminus \mathcal{A}} \sqsubseteq \sigma$.

Definition 6. *A* Concurrent Game Model *(CGM) over Σ and Π with root s_0 is a tuple $\mathcal{M} = (\mathcal{F}, \Pi, \pi)$, where $\mathcal{F} = (\Sigma, \mathcal{S}, s_0, d, \delta)$ is a CGF; Π is the set of propositional symbols; and $\pi : \mathcal{S} \longrightarrow 2^{\Pi}$ is a valuation function.*

Definition 7. *Let $\mathcal{M} = (\Sigma, \mathcal{S}, s_0, d, \delta, \Pi, \pi)$ be a CGM with $s \in \mathcal{S}$. The satisfaction relation, denoted by $\models$, is inductively defined as follows.*

- *$\langle \mathcal{M}, s \rangle \models p$ iff $p \in \pi(s)$, for all $p \in \Pi$;*
- *$\langle \mathcal{M}, s \rangle \models \neg\varphi$ iff $\langle \mathcal{M}, s \rangle \not\models \varphi$;*
- *$\langle \mathcal{M}, s \rangle \models (\varphi \rightarrow \psi)$ iff $\langle \mathcal{M}, s \rangle \models \varphi$ implies $\langle \mathcal{M}, s \rangle \models \psi$;*
- *$\langle \mathcal{M}, s \rangle \models \bigwedge_{i=1}^{n} \varphi_i$ iff $\langle \mathcal{M}, s \rangle \models \varphi_i$ for all i, $1 \le i \le n$;*
- *$\langle \mathcal{M}, s \rangle \models \bigvee_{i=1}^{n} \varphi_i$ iff $\langle \mathcal{M}, s \rangle \models \varphi_i$ for some i, $1 \le i \le n$;*
- *$\langle \mathcal{M}, s \rangle \models [\mathcal{A}]\varphi$ iff there exists an $\mathcal{A}$-move $\sigma_{\mathcal{A}} \in D(\mathcal{A}, s)$ s.t.*
  for all $\sigma \in D(s)$ $\sigma_{\mathcal{A}} \sqsubseteq \sigma$ implies $\langle \mathcal{M}, \delta(s, \sigma) \rangle \models \varphi$;
- *$\langle \mathcal{M}, s \rangle \models \langle \mathcal{A} \rangle\varphi$ iff for all $\mathcal{A}$-moves $\sigma_{\mathcal{A}} \in D(\mathcal{A}, s)$*
  exists $\sigma \in D(s)$ s.t. $\sigma_{\mathcal{A}} \sqsubseteq \sigma$ and $\langle \mathcal{M}, \delta(s, \sigma) \rangle \models \varphi$.

Definition 8. *Let $\mathcal{M}$ be a CGM. A* CL *formula φ is* satisfied at the state s in *$\mathcal{M}$ if $\langle \mathcal{M}, s \rangle \models \varphi$ and φ is* satisfiable in $\mathcal{M}$, *denoted by $\mathcal{M} \models \varphi$, if $\langle \mathcal{M}, s_0 \rangle \models \varphi$. A finite set $\Phi \subset \mathsf{WFF}_{\mathsf{CL}}$ is* satisfiable in a state s in $\mathcal{M}$, *denoted by $\langle \mathcal{M}, s \rangle \models \Phi$, if for all $\varphi_i \in \Phi$, $0 \le i \le n$, $\langle \mathcal{M}, s \rangle \models \varphi_i$, and Φ is* satisfiable in $\mathcal{M}$, *denoted by $\mathcal{M} \models \Phi$, if $\langle \mathcal{M}, s_0 \rangle \models \Phi$.*

As discussed in [11,18,23] three different notions of satisfiability emerge from the relation between the set of agents occurring in a formula and the set of agents in the language. It turns out that all those notions of satisfiability can be reduced to *tight satisfiability* [23]. We denote by $\Sigma_{\varphi} \subseteq \Sigma$, the set of agents occurring in a well-formed formula φ or the set $\{a\}$ for some arbitrary agent $a \in \Sigma$ if the set of agents occurring in φ is empty. If Φ is a set of well-formed formulae, $\Sigma_{\Phi} \subseteq \Sigma$ denotes $\bigcup_{\varphi \in \Phi} \Sigma_{\varphi}$.

Definition 9 (Tight satisfiability). *A* CL *formula φ is* satisfiable *if there is a Concurrent Game Model $\mathcal{M} = (\Sigma_{\varphi}, \mathcal{S}, s_0, d, \delta, \Pi, \pi)$ such that $\langle \mathcal{M}, s_0 \rangle \models \varphi$. A finite set Φ of* CL *formulae is* satisfiable, *if there is a CGM $\mathcal{M}$ over Σ_{Φ} and Π with root s_0 such that $\langle \mathcal{M}, s_0 \rangle \models \Phi$.*

3 Unrefined Resolution for CL

The resolution method presented in [16] proceeds by translating a CL formula φ that is to be tested for (un)satisfiability into a clausal normal form $\mathcal{C}$, a *coalition problem in Divided Separated Normal Form for Coalition Logic* ($\mathsf{DSNF}_{\mathsf{CL}}$), to which then resolution-based inference rules are applied. The application of these rules always terminates, either resulting in a coalition problem $\mathcal{C}'$ that is evidently contradictory or, otherwise, satisfiable. The formula φ is satisfiable iff $\mathcal{C}'$ is satisfiable.

Definition 10. *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* $\mathsf{WFF}_{\mathsf{CL}}$*; 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. Formally, the semantics of coalition problems is defined as follows.

Definition 11. *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* $\mathcal{M} = (\Sigma_{\mathcal{C}}, \mathcal{S}, s_0, d, \delta, \Pi, \pi)$ *is a CGM, then* $\mathcal{M} \models \mathcal{C}$ *if, and only if,* $\langle \mathcal{M}, s_0 \rangle \models \mathcal{I}$ *and* $\langle \mathcal{M}, s \rangle \models \mathcal{U} \cup \mathcal{N}$*, for all* $s \in \mathcal{S}$*. 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*.

Definition 12. *A* coalition problem in $\mathsf{DSNF}_{\mathsf{CL}}$ *is a coalition problem* $(\mathcal{I},\mathcal{U},\mathcal{N})$ *such that* $\mathcal{I}$*, the set of* initial clauses*, and* $\mathcal{U}$*, the set of* global clauses*, are finite sets of propositional clauses* $\bigvee_{j=1}^{n} l_j$*, and* $\mathcal{N}$*, the set of* coalition clauses*, is a finite set of formulae in* $\mathsf{WFF}_{\mathsf{CL}}$ *of the form* $\bigwedge_{i=1}^{m} l'_i \to [\mathcal{A}] \bigvee_{j=1}^{n} l_j$ *(*positive coalition clauses*) or* $\bigwedge_{i=1}^{m} l'_i \to \langle \mathcal{A} \rangle \bigvee_{j=1}^{n} l_j$ *(*negative coalition clauses*), where* $m, n \geq 0$ *and* l'_i, l_j*, for all* $1 \leq i \leq m$*,* $1 \leq j \leq n$*, are literals such that within every conjunction and every disjunction literals are pairwise different.*

Definition 13. *A* coalition problem in unit $\mathsf{DSNF}_{\mathsf{CL}}$ $(\mathcal{I},\mathcal{U},\mathcal{N})$ *is a coalition problem in* $\mathsf{DSNF}_{\mathsf{CL}}$ *such that coalition clauses in* $\mathcal{N}$ *have the following forms (where* p *is a propositional symbol):*

positive coalition clauses	$\bigwedge_{i=1}^{m} l'_i \to [\mathcal{A}]p$
negative coalition clauses	$\bigwedge_{i=1}^{m} l'_i \to \langle \mathcal{A} \rangle p$

The transformation of a CL formula φ into a coalition problem in $\mathsf{DSNF}_{\mathsf{CL}}$ or unit $\mathsf{DSNF}_{\mathsf{CL}}$ uses a set of rewrite rules that transforms φ into negation normal form, removes propositionally redundant subformulae, and uses renaming [19] in order to bring coalition problems closer to $\mathsf{DSNF}_{\mathsf{CL}}$. For a description of the transformation rules and proofs of the following theorem see [16,25].

$$\textsf{IRES1}\quad \frac{\begin{array}{ll} C \vee l & \in \mathcal{I} \\ D \vee \neg l & \in \mathcal{I} \cup \mathcal{U} \end{array}}{\begin{array}{ll} C \vee D & \in \mathcal{I} \end{array}} \qquad \textsf{GRES1}\quad \frac{\begin{array}{ll} C \vee l & \in \mathcal{U} \\ D \vee \neg l & \in \mathcal{U} \end{array}}{\begin{array}{ll} C \vee D & \in \mathcal{U} \end{array}}$$

$$\begin{array}{l}\textsf{CRES1}\\ \mathcal{A} \cap \mathcal{B} = \emptyset\end{array}\quad \frac{\begin{array}{rl} P \to [\mathcal{A}](C \vee l) & \in \mathcal{N} \\ Q \to [\mathcal{B}](D \vee \neg l) & \in \mathcal{N} \end{array}}{\begin{array}{rl} P \wedge Q \to [\mathcal{A} \cup \mathcal{B}](C \vee D) & \in \mathcal{N} \end{array}} \qquad \textsf{CRES2}\quad \frac{\begin{array}{rl} C \vee l & \in \mathcal{U} \\ Q \to [\mathcal{A}](D \vee \neg l) & \in \mathcal{N} \end{array}}{\begin{array}{rl} Q \to [\mathcal{A}](C \vee D) & \in \mathcal{N} \end{array}}$$

$$\begin{array}{l}\textsf{CRES3}\\ \mathcal{A} \subseteq \mathcal{B}\end{array}\quad \frac{\begin{array}{rl} P \to [\mathcal{A}](C \vee l) & \in \mathcal{N} \\ Q \to \langle\mathcal{B}\rangle(D \vee \neg l) & \in \mathcal{N} \end{array}}{\begin{array}{rl} P \wedge Q \to \langle\mathcal{B} \setminus \mathcal{A}\rangle(C \vee D) & \in \mathcal{N} \end{array}} \qquad \textsf{CRES4}\quad \frac{\begin{array}{rl} C \vee l & \in \mathcal{U} \\ Q \to \langle\mathcal{A}\rangle(D \vee \neg l) & \in \mathcal{N} \end{array}}{\begin{array}{rl} Q \to \langle\mathcal{A}\rangle(C \vee D) & \in \mathcal{N} \end{array}}$$

$$\textsf{RW1}\quad \frac{\bigwedge_{i=1}^{n} l_i \to [\mathcal{A}]\mathbf{false} \in \mathcal{N}}{\bigvee_{i=1}^{n} \neg l_i \quad \in \mathcal{U}} \qquad \textsf{RW2}\quad \frac{\bigwedge_{i=1}^{n} l_i \to \langle\mathcal{A}\rangle\mathbf{false} \in \mathcal{N}}{\bigvee_{i=1}^{n} \neg l_i \quad \in \mathcal{U}}$$

where $(\mathcal{I}, \mathcal{U}, \mathcal{N})$ is a coalition problem in $\mathsf{DSNF_{CL}}$; P, Q are conjunctions of literals; C, D are disjunctions of literals; l, l_i are literals; and $\mathcal{A}, \mathcal{B} \subseteq \Sigma$ are coalitions.

Figure 1. Resolution Calculus $\mathsf{RES_{CL}}$

Theorem 1 (Preservation of satisfiability). *Let φ be a $\mathsf{WFF_{CL}}$. Then we can compute in polynomial time a coalition problem $\mathcal{C}$ in $\mathsf{DSNF_{CL}}$ or a coalition problem $\mathcal{C}'$ in unit $\mathsf{DSNF_{CL}}$ such that $\mathcal{C}$ and $\mathcal{C}'$ are satisfiable iff φ is satisfiable.*

The resolution calculus $\mathsf{RES_{CL}}$, introduced in [16], consists of the inference rules shown in Figure 1.

Theorem 2 (Soundness, Completeness, Termination). *Let $\mathcal{C}$ be a coalition problem in unit $\mathsf{DSNF_{CL}}$. Then any derivation from $\mathcal{C}$ by $\mathsf{RES_{CL}}$ terminates and there is a refutation for $\mathcal{C}$ using the inference rules* **IRES1***,* **GRES1***,* **CRES1–4***, and* **RW1–2** *iff $\mathcal{C}$ is unsatisfiable.*

This corrects the completeness result stated in [16] which claims completeness of $\mathsf{RES_{CL}}$ for coalition problems in $\mathsf{DSNF_{CL}}$ instead of unit $\mathsf{DSNF_{CL}}$[4].

4 Ordered Resolution for CL

Ordering refinements are a commonly used approach to reducing the search space of resolution for classical propositional and first-order logic. They are utilised by all state-of-the-art resolution-based theorem provers for first-order logic, including E [21], SPASS [24], and Vampire [15]. Ordering refinements have also been used in the context of hybrid, modal and temporal logics including H(@) [3], PLTL [14] and CTL [25].

An *atom ordering* is a well-founded and total ordering $\succ$ on the set Π. The ordering $\succ$ is extended to literals such that for each $p \in \Pi$, $\neg p \succ p$, and for each

[4] Proofs for all results in this paper can be found in `http://cgi.csc.liv.ac.uk/~ullrich/publications/Tableaux2015proofs.pdf`.

$q \in \Pi$ such that $q \succ p$ then $q \succ \neg p$ and $\neg q \succ \neg p$. A literal l is *maximal* with respect to a propositional disjunction C iff for every literal l' in C, $l' \not\succ l$.

We could use the ordering $\succ$ to restrict the applicability of the rules **IRES1**, **GRES1**, **CRES1** to **CRES4** so that a rule is only applicable if and only if the literal l in $C \vee l$ is maximal with respect C and the literal $\neg l$ in $D \vee \neg l$ is maximal with respect to D. One would normally expect that the calculus we obtain by way of this restriction is complete for any ordering, see, for example [4,13,14,25].

However, it turns out that such a restriction would render our calculus incomplete. Consider the following example, a coalition problem corresponding to the unsatisfiable CL formula $[1](p \wedge \neg p)$:

1.	t_0	$[\mathcal{I}]$	3.	$\neg t_1 \vee \neg p$	$[\mathcal{U}]$
2.	$\neg t_1 \vee p$	$[\mathcal{U}]$	4.	$t_0 \rightarrow [1]t_1$	$[\mathcal{N}]$

Assume that the ordering on propositional symbols is $t_0 \succ t_1 \succ p$. Then the only inferences possible are the following:

5.	$t_0 \rightarrow [1]p$	$[\mathcal{N}$, **CRES2**, $2, 4, t_1]$	6.	$t_0 \rightarrow [1]\neg p$	$[\mathcal{N}$, **CRES2**, $3, 4, t_1]$

A resolution inference using **CRES1** with Clauses (5) and (6) as premises is not possible as the sets of agents in the two clauses is not disjoint. Using the unrefined calculus $\mathsf{RES_{CL}}$ or using a different ordering, namely, $p \succ t_1 \succ t_0$ allows us to construct a refutation for this example:

5'.	$\neg t_1$	$[\mathcal{U}$, **GRES1**, $2, 3, p]$	7'.	$\neg t_0$	$[\mathcal{U}$, **RW1**, 6'$]$
6'.	$t_0 \rightarrow [1]$**false**	$[\mathcal{N}$, **CRES2**, 4,5'$, t_1]$	8'.	**false**	$[\mathcal{U}$, **IRES1**, 1,7'$, t_0]$

Note that if we were to use other refinements of resolution that are not consequence complete to restrict the applicability of the rules of $\mathsf{RES_{CL}}$, then this would also result in an incomplete calculus. For example, instead of an ordering refinement, we could use a *selection function* [4] to restrict inferences on clauses (2) and (3) to the negative literal $\neg t_1$. Then again the only clauses immediately derivable from the clauses (1) to (4) are the clauses (5) and (6), with no further inferences being possible.

The incompleteness of a naive ordering refinement of $\mathsf{RES_{CL}}$ is related to the fact that a derived clause does not accurately reflect the constraints on the agents' moves inherited from the premises of a resolution inference. In order to overcome this problem we need a better representation of these constraints, essentially we need to skolemize implicitly existentially quantified variables in coalition modalities. To this end we introduce *Vector Coalition Logic* incorporating the notions of *coalition vector* and use these vectors to replace coalition modalities in coalition clauses.

Definition 14. *Let* $|\Sigma| = k$. *A* coalition vector $\vec{c}$ *is a k-tuple* $(m_1, \ldots, m_k)$ *such that for every* a, $1 \leq a \leq k$, $\vec{c}[a] = m_a$, *the component with index* a, *is either an integer number not equal to zero or the symbol* $*$ *and for every* a, a', $1 \leq a < a' \leq k$, *if* $\vec{c}[a] < 0$ *and* $\vec{c}[a'] < 0$ *then* $\vec{c}[a] = \vec{c}[a']$.

A coalition vector $\vec{c}$ *is* negative *if* $\vec{c}[a] < 0$ *for some* a, $1 \leq a \leq k$. *Otherwise,* $\vec{c}$ *is* positive. *We denote by* $\vec{c}^{\,+}$ *that* $\vec{c}$ *is positive and by* $\vec{c}^{\,-}$ *that* $\vec{c}$ *is negative.*

For example, given $\Sigma = \{1, \ldots, 6\}$, $\vec{c}_1 = (1, *, *, 3, 1, *)$, $\vec{c}_2 = (*, -2, *, 3, *, -2)$, and $\vec{c}_3 = (1, -2, *, 3, *, -2)$ are coalition vectors, $\vec{c}_1$ is positive, while $\vec{c}_2$ and $\vec{c}_3$ are negative.

Definition 15. *The set $\mathsf{WFF}_{\mathsf{VCL}}$ of* Vector Coalition Logic (VCL) formulae *is inductively defined as follows.*
– *if p is a propositional symbol in Π, then p and $\neg p$ are VCL formulae;*
– *if φ is a propositional formula and ψ is a VCL formula, then $(\varphi \to \psi)$ is a VCL formula;*
– *if φ_i, $1 \leq i \leq n$, $n \in \mathbb{N}_0$, are VCL formula, then so are $(\varphi_1 \wedge \ldots \wedge \varphi_n)$, also written $\bigwedge_{i=1}^{n} \varphi_i$, and $(\varphi_1 \vee \ldots \vee \varphi_n)$, also written $\bigvee_{i=1}^{n} \varphi_i$; and*
– *if $\vec{c}$ is a coalition vector and φ is a VCL formula, then so is $\vec{c}\varphi$.*

Note that negation is restricted to propositional symbols. In particular, we do not allow formulae of the form $\neg\vec{c}\varphi$. Since such formulae do not occur in our normal form, this is not a restriction for our purposes.

In order to define the semantics of $\mathsf{WFF}_{\mathsf{VCL}}$ formulae we can reuse Concurrent Game Frames, but need to extend Concurrent Game Models with *choice functions* that give meaning to coalition vectors.

Definition 16. *A* Concurrent Game Model with Choice Functions *(CGM_{CF}) is a tuple $\mathcal{M} = (\mathcal{F}, \Pi, \pi, F^+, F^-)$, where*

– *$\mathcal{F} = (\Sigma, \mathcal{S}, s_0, d, \delta)$ is a CGF;*
– *Π is the set of propositional symbols;*
– *$\pi : \mathcal{S} \longrightarrow 2^{\Pi}$ is a valuation function;*
– *$F^+ = \{f^i \mid i \in \mathbb{N}\}$ is a set of functions such that $f^i : \mathcal{S} \times \Sigma \longrightarrow \mathbb{N}$ and $f^i(s, a) \in D(a, s)$ for every $i \in \mathbb{N}$, $a \in \Sigma$, and $s \in \mathcal{S}$;*
– *$F^- = \{g^i_n \mid i, n \in \mathbb{N}$ and $n \leq |\Sigma|\}$ is a set of functions such that $g^i_n : \mathcal{S} \times \Sigma \times \mathbb{N}^n \longrightarrow \mathbb{N}$ and $g^i_n(s, a, (m_1, \ldots, m_n)) \in D(a, s)$ for every $i \in \mathbb{N}$, $a \in \Sigma$, $s \in \mathcal{S}$, $(m_1, \ldots, m_n) \in \mathbb{N}^n$.*

Definition 17. *Let $\mathcal{M} = (\mathcal{F}, \Pi, \pi, F^+, F^-)$ be a CGM_{CF} and let s be a state in $\mathcal{S}$. Let $\vec{c}$ be a coalition vector where $\{a \mid 1 \leq a \leq |\Sigma| \wedge (\vec{c}[a] > 0 \vee \vec{c}[a] = *)\} = \{a_1, \ldots, a_n\}$ with $a_1 < \cdots < a_n$. A move vector σ* instantiates *the coalition vector $\vec{c}$ at state s, denoted by $\vec{c} \sqsubseteq \sigma$, if:*
– *$\sigma(a') = f^{\vec{c}[a']}(s, a')$ for each a', $1 \leq a' \leq |\Sigma|$ and $\vec{c}[a'] > 0$,*
– *$\sigma(a) = g_n^{|\vec{c}[a]|}(s, a, (\sigma(a_1), \ldots, \sigma(a_n)))$ for each a, $1 \leq a \leq |\Sigma|$ and $\vec{c}[a] < 0$.*

The intuition underlying Definition 17 is the following. A coalition vector such as, for example, $(1, -2, *, 3, *, -2)$, indicates that agents 1 and 4 are committed to moves m_1 and m_4 that depend only on the state s they are currently in and are determined by the choice functions f^1 and f^3: $m_1 = f^1(s, 1)$ and $m_4 = f^3(s, 4)$, respectively. Agents 3 and 5 will perform arbitrary moves m_3 and m_5 of their choice in s. Finally, agents 2 and 6 will choose their moves m_2 and m_6 in reaction to the moves of all the other four agents and their moves are determined by the choice function $g_4^{|-2|} = g_4^2 : m_2 = g_4^2(s, 2, (m_1, m_3, m_4, m_5))$ and $m_6 = g_4^2(s, 6, (m_1, m_3, m_4, m_5))$, respectively.

Definition 18. *Let $\mathcal{M} = (\mathcal{F}, \Pi, \pi, F^+, F^-)$ be a CGM$_{\text{CF}}$ with $s \in \mathcal{S}$. The satisfaction relation $\models$ between $\mathcal{M}$, s and* VCL *formulae is inductively defined as follows.*

– $\langle \mathcal{M}, s\rangle \models p$ *iff* $p \in \pi(s)$, *for all* $p \in \Pi$;
– $\langle \mathcal{M}, s\rangle \models \neg\varphi$ *iff* $\langle \mathcal{M}, s\rangle \not\models \varphi$;
– $\langle \mathcal{M}, s\rangle \models (\varphi \to \psi)$ *iff* $\langle \mathcal{M}, s\rangle \models \varphi$ *implies* $\langle \mathcal{M}, s\rangle \models \psi$;
– $\langle \mathcal{M}, s\rangle \models \bigwedge_{i=1}^{n} \varphi_i$ *iff* $\langle \mathcal{M}, s\rangle \models \varphi_i$ *for all* i, $1 \leq i \leq n$;
– $\langle \mathcal{M}, s\rangle \models \bigvee_{i=1}^{n} \varphi_i$ *iff* $\langle \mathcal{M}, s\rangle \models \varphi_i$ *for some* i, $1 \leq i \leq n$;
– $\langle \mathcal{M}, s\rangle \models \vec{c}\varphi$ *iff for all* $\sigma \in D(s)$, $\vec{c} \sqsubseteq \sigma$ *implies* $\langle \mathcal{M}, \delta(s, \sigma)\rangle \models \varphi$.

The notions of satisfiability of a VCL formula and a set of VCL formulae are defined as in Definitions 8 and 9 but with respect to CGM$_{\text{CF}}$'s instead of CGM's.

We can now present a normal form for Coalition Logic using VCL formulae:

Definition 19. *A* coalition problem in $\mathsf{DSNF}_{\mathsf{VCL}}$ *is a tuple* $(\mathcal{I}, \mathcal{U}, \mathcal{N})$ *where* $\mathcal{I}$ *is a set of* initial clauses*,* $\mathcal{U}$ *is a set of* global clauses*, and* $\mathcal{N}$*, the set of* coalition clauses*, consists of* VCL *formulae of the form* $\bigwedge_{i=1}^{m} l'_i \to \vec{c} \bigvee_{j=1}^{n} l_j$ *where* $m, n \geq 0$ *and* l'_i, l_j*, for all* $1 \leq i \leq m$*,* $1 \leq j \leq n$*, are literals such that within every conjunction and every disjunction literals are pairwise different, and* $\vec{c}$ *is a coalition vector.*

The notion of satisfiability of a coalition problem in $\mathsf{DSNF}_{\mathsf{VCL}}$ is defined as in Definition 11 but with respect to CGM$_{\text{CF}}$'s instead of CGM's.

Given a coalition problem $\mathcal{C}$ in $\mathsf{DSNF}_{\mathsf{CL}}$ we can obtain a coalition problem in $\mathsf{DSNF}_{\mathsf{VCL}}$ by exhaustive application of the following two rewrite rules:

$\tau_{[\mathcal{A}]}$ $(\mathcal{I}, \mathcal{U}, \mathcal{N} \cup \{t \to [\mathcal{A}]\psi\}) \Rightarrow_{[\_]} (\mathcal{I}, \mathcal{U}, \mathcal{N} \cup \{t \to \vec{c}^{\,i}_{\mathcal{A}}\psi\})$
where ψ is a disjunction of literals, i is a natural number not occurring as an index of some coalition vector in $\mathcal{N}$, and $\vec{c}^{\,i}_{\mathcal{A}}$ is a coalition vector such that $\vec{c}^{\,i}_{\mathcal{A}}(a) = i$ for every $a \in \mathcal{A}$ and $\vec{c}^{\,i}_{\mathcal{A}}(a') = *$ for every $a' \notin \mathcal{A}$.

$\tau_{\langle\mathcal{A}\rangle}$ $(\mathcal{I}, \mathcal{U}, \mathcal{N} \cup \{t \to \langle\mathcal{A}\rangle\psi\}) \Rightarrow_{\langle\_\rangle} (\mathcal{I}, \mathcal{U}, \mathcal{N} \cup \{t \to \vec{c}^{\,-i}_{\mathcal{A}}\psi\})$
where ψ is a disjunction of literals, i is a natural number not occurring as an index of some coalition vector in $\mathcal{N}$, and $\vec{c}^{\,-i}_{\mathcal{A}}$ is a coalition vector such that $\vec{c}^{\,-i}_{\mathcal{A}}(a') = -i$ for every $a' \notin \mathcal{A}$ and $\vec{c}^{\,-i}_{\mathcal{A}}(a) = *$ for every $a \in \mathcal{A}$.

Theorem 3. *Let $\mathcal{C}$ be a coalition problem in $\mathsf{DSNF}_{\mathsf{CL}}$ and let $\mathcal{C}'$ be obtained by exhaustively applying the rewrite rules $\tau_{[\mathcal{A}]}$ and $\tau_{\langle\mathcal{A}\rangle}$ to $\mathcal{C}$. Then $\mathcal{C}'$ is a coalition problem in $\mathsf{DSNF}_{\mathsf{VCL}}$ and $\mathcal{C}'$ is satisfiable if and only if $\mathcal{C}$ is satisfiable.*

Before we can present the inference rules for coalition problems in $\mathsf{DSNF}_{\mathsf{VCL}}$, we need to define when two coalition vectors are 'unifiable'. To this end we introduce the notion of a *merge* of two coalition vectors.

Definition 20. *Let $\vec{c}_1$ and $\vec{c}_2$ be two coalition vectors of length n. The coalition vector $\vec{c}_2$ is an* instance *of $\vec{c}_1$ and $\vec{c}_1$ is* more general than *$\vec{c}_2$, written $\vec{c}_1 \sqsubseteq \vec{c}_2$, if $\vec{c}_2[i] = \vec{c}_1[i]$ for every i, $1 \leq i \leq n$, with $\vec{c}_1[i] \neq *$. We say that a coalition vector $\vec{c}_3$ is a* common instance *of $\vec{c}_1$ and $\vec{c}_2$ if $\vec{c}_3$ is an instance*

$$\textbf{IRES1}\quad \begin{array}{ll} C\lor l & \in\mathcal{I}\\ D\lor\neg l & \in \mathcal{I}\cup\mathcal{U}\\ \hline C\lor D & \in\mathcal{I}\end{array} \qquad\qquad \textbf{GRES1}\quad \begin{array}{ll} C\lor l & \in\mathcal{U}\\ D\lor\neg l & \in \mathcal{U}\\ \hline C\lor D & \in\mathcal{U}\end{array}$$

$$\textbf{VRES1}\quad \begin{array}{ll} P\rightarrow\vec{c}_1(C\lor l) & \in\mathcal{N}\\ Q\rightarrow\vec{c}_2(D\lor\neg l) & \in\mathcal{N}\\ \hline P\land Q\rightarrow\vec{c}_1\downarrow\vec{c}_2(C\lor D) & \in\mathcal{N}\end{array} \qquad \textbf{VRES2}\quad \begin{array}{ll} C\lor l & \in\mathcal{U}\\ Q\rightarrow\vec{c}(D\lor\neg l) & \in\mathcal{N}\\ \hline Q\rightarrow\vec{c}(C\lor D) & \in\mathcal{N}\end{array}$$

$$\textbf{RW}\quad \begin{array}{ll} \bigwedge_{i=1}^{n} l_i\rightarrow\vec{c}\,\mathbf{false} & \in\mathcal{N}\\ \hline \bigvee_{i=1}^{n}\neg l_i & \in\mathcal{U}\end{array}$$

where $(\mathcal{I},\mathcal{U},\mathcal{N})$ is a coalition problem in $\mathsf{DSNF}_{\mathsf{CL}}$; P, Q are conjunctions of literals; C, D are disjunctions of literals; l, l_i are literals; $\vec{c}$, $\vec{c}_1$, $\vec{c}_2$ are coalition vectors; in **VRES1**, $\vec{c}_1$ and $\vec{c}_2$ are mergeable; and in **IRES1**, **GRES1**, **VRES1** and **VRES2**, l is maximal with respect to C and $\neg l$ is maximal with respect to D.

Figure 2. Resolution Calculus $\mathsf{RES}^{\succ}_{\mathsf{CL}}$

of both $\vec{c}_1$ and $\vec{c}_2$. A coalition vector $\vec{c}_3$ is a merge *of $\vec{c}_1$ and $\vec{c}_2$ if $\vec{c}_3$ is a common instance of $\vec{c}_1$ and $\vec{c}_2$, and for any common instance $\vec{c}_4$ of $\vec{c}_1$ and $\vec{c}_2$ we have $\vec{c}_3 \sqsubseteq \vec{c}_4$. If there exists a merge for two coalition vectors $\vec{c}_1$ and $\vec{c}_2$ then we say that $\vec{c}_1$ and $\vec{c}_2$ are* mergeable. *We denote the merge of $\vec{c}_1$ and $\vec{c}_2$ by $\vec{c}_1\downarrow\vec{c}_2$ and write $\vec{c}_1\downarrow\vec{c}_2 = \mathbf{undef}$ if $\vec{c}_1$ and $\vec{c}_2$ are not mergeable.*

For example, the merge of $\vec{c}_1 = (1,*,*,3,1,*)$, $\vec{c}_2 = (*,-2,*,3,*,-2)$ is $\vec{c}_4 = (1,-2,*,3,1,-2)$, while $\vec{c}_1$ and $\vec{c}_5 = (1,*,*,4,1,*)$ are not mergeable nor are $\vec{c}_2$ and $\vec{c}_6 = (-5,*,*,3,*,*)$.

Remark 1. Let $\vec{c}_1$ and $\vec{c}_2$ be two coalition vectors. If there is a common instance $\vec{c}_3$ of $\vec{c}_1$ and $\vec{c}_2$ then there exists a merge of $\vec{c}_1$ and $\vec{c}_2$.

The resolution calculus $\mathsf{RES}^{\succ}_{\mathsf{CL}}$, where $\succ$ is an atom ordering, consists of the inference rules shown in Figure 2. Note that **VRES1** and **VRES2** in $\mathsf{RES}^{\succ}_{\mathsf{CL}}$, just as **CRES1** to **CRES4** in $\mathsf{RES}_{\mathsf{CL}}$, do not allow to resolve on literals on the left-hand side of an implication in a coalition clause.

Definition 21. *A* derivation *from a coalition problem $\mathcal{C}$ in $\mathsf{DSNF}_{\mathsf{VCL}}$ by $\mathsf{RES}^{\succ}_{\mathsf{CL}}$ is a sequence $\mathcal{C}_0,\mathcal{C}_1,\mathcal{C}_2,\ldots$ of coalition problems in $\mathsf{DSNF}_{\mathsf{VCL}}$ such that $\mathcal{C}_0 = \mathcal{C}$, $\mathcal{C}_i = (\mathcal{I}_i,\mathcal{U}_i,\mathcal{N}_i)$, and $\mathcal{C}_{i+1}$ is either*
– *$(\mathcal{I}_i\cup\{E\},\mathcal{U}_i,\mathcal{N}_i)$, where E is a conclusion of* **IRES1***;*
– *$(\mathcal{I}_i,\mathcal{U}_i\cup\{E\},\mathcal{N}_i)$, where E is a conclusion of* **GRES1** *or* **RW***; or*
– *$(\mathcal{I}_i,\mathcal{U}_i,\mathcal{N}_i\cup\{E\})$, where E is a conclusion of* **VRES1** *or* **VRES2***.*

A refutation *of $\mathcal{C}_0$ by $\mathsf{RES}^{\succ}_{\mathsf{CL}}$ is a derivation $\mathcal{C}_0,\ldots,\mathcal{C}_n = (\mathcal{I}_n,\mathcal{U}_n,\mathcal{N}_n)$ from $\mathcal{C}_0$ by $\mathsf{RES}^{\succ}_{\mathsf{CL}}$ such that* $\mathbf{false}\in\mathcal{I}_n\cup\mathcal{U}_n$*.*

We return to our previous example on page 7. The corresponding coalition problem in $\mathsf{DSNF}_{\mathsf{VCL}}$ consist of the following clauses:

1.	t_0	$[\mathcal{I}]$	3.	$\neg t_1\lor\neg p$	$[\mathcal{U}]$
2.	$\neg t_1\lor p$	$[\mathcal{U}]$	4.	$t_0\rightarrow(1)t_1$	$[\mathcal{N}]$

Note that the number 1 in the vector (1) does not refer to agent 1, but to a specific move by agent 1. Assume again that the ordering on propositional symbols is $t_0 \succ t_1 \succ p$. Then a refutation using $\mathsf{RES}^{\succ}_{\mathsf{CL}}$ proceeds as follows:

5.	$t_0 \rightarrow (1)p$	$[\mathcal{N}, \textbf{VRES2}, 2, 4, t_1]$	8.	$\neg t_0$	$[\mathcal{U}, \textbf{RW}, 7]$
6.	$t_0 \rightarrow (1)\neg p$	$[\mathcal{N}, \textbf{VRES2}, 3, 4, t_1]$	9.	**false**	$[\mathcal{I}, \textbf{IRES1}, 1, 8, t_0]$
7.	$t_0 \rightarrow (1)\textbf{false}$	$[\mathcal{N}, \textbf{VRES1}, 5, 6, p]$			

The derivation of Clause (7) is the crucial step as it takes advantage of the fact that $(1)p$ and $(1)\neg p$ can be resolved by $\mathsf{RES}^{\succ}_{\mathsf{CL}}$ while $[1]p$ and $[1]\neg p$ cannot be resolved by $\mathsf{RES}_{\mathsf{CL}}$.

Soundness. Soundness of the inference rules of $\mathsf{RES}^{\succ}_{\mathsf{CL}}$ is shown model-theoretically: For each rule we show that if the premises of an application of the rule have a model $\mathcal{M}$, then $\mathcal{M}$ is also a model for the conclusion of that application.

Theorem 4 (Soundness of $\mathsf{RES}^{\succ}_{\mathsf{CL}}$). *Let $\mathcal{C}$ be a coalition problem in $\mathsf{DSNF}_{\mathsf{VCL}}$. Let $\mathcal{C}'$ be the coalition problem in $\mathsf{DSNF}_{\mathsf{VCL}}$ obtained from $\mathcal{C}$ by applying any of the inference rules* **IRES1***,* **GRES1***,* **VRES1***,* **VRES2** *and* **RW** *to $\mathcal{C}$. If $\mathcal{C}$ is satisfiable, then $\mathcal{C}'$ is satisfiable.*

Termination. Consider a derivation from the coalition problem $\mathcal{C}$. The set of propositional symbols $\Pi_{\mathcal{C}}$ occurring in $\mathcal{C}$ is finite and the inference rules do not introduce new propositional symbols, so the number of possible literals is finite. We keep propositional conjunctions and disjunction in simplified form, so there are only finitely many that may occur in any clause. Also, in $\mathcal{C}$ only a finite set $I_{\mathcal{C}} \subset \mathbb{Z}$ of numbers occurs in coalition vectors and all coalition vectors in $\mathcal{C}$ have the same length, say, k. Then the number of coalition vectors that may occur in a derivation is bounded by $(|I_{\mathcal{C}}| + 1)^k$. 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 5. *Let $\mathcal{C} = (\mathcal{I}, \mathcal{U}, \mathcal{N})$ be a coalition problem in $\mathsf{DSNF}_{\mathsf{VCL}}$. Then any derivation from $\mathcal{C}$ by $\mathsf{RES}^{\succ}_{\mathsf{CL}}$ terminates.*

Completeness. In our completeness proof for $\mathsf{RES}^{\succ}_{\mathsf{CL}}$ we show that a refutation of a CL formula φ by Goranko and Shkatov's tableau procedure T_{ATL} for ATL [11] can be used to guide the construction of a refutation of the coalition problem $\mathcal{C}_\varphi$ corresponding to φ by $\mathsf{RES}^{\succ}_{\mathsf{CL}}$. The tableau procedure proceeds in two phases, a construction phase in which a graph structure for φ is build, and an elimination phase in which parts of the graph that cannot be used to create a CGM for φ are deleted. The formula φ is satisfiable iff at the end of the elimination phase a non-empty graph with a node containing φ remains. In the proof, we first define a mapping $\mathrm{tr}_{\mathcal{C}_\varphi}$ of coalition problems in $\mathsf{DSNF}_{\mathsf{VCL}}$ to ATL formulae. Second, we show that if in the graph G constructed for $\mathrm{tr}_{\mathcal{C}_\varphi}(\mathcal{C})$ by T_{ATL} there is an elimination step possible, then we can derive from $\mathcal{C}$ a coalition problem $\mathcal{C}'$ by $\mathsf{RES}^{\succ}_{\mathsf{CL}}$ such that the graph G' constructed for $\mathrm{tr}_{\mathcal{C}_\varphi}(\mathcal{C}')$ by T_{ATL} is a sub-graph of G. By induction we can then show that if a sequence of elimination steps by T_{ATL} produces an empty graph, then the corresponding derivation by $\mathsf{RES}^{\succ}_{\mathsf{CL}}$ is a refutation of $\mathcal{C}_\varphi$.

Theorem 6. *Let $\varphi \in \mathit{WFF}_{\mathsf{CL}}$ and $\mathcal{C} = (\mathcal{I}, \mathcal{U}, \mathcal{N})$ be the corresponding coalition problem in $\mathsf{DSNF}_{\mathsf{VCL}}$. If φ is unsatisfiable then there is a refutation of $\mathcal{C}$ by $\mathsf{RES}^{\succ}_{\mathsf{CL}}$.*

Complexity. The satisfiability problem for CL is PSPACE-complete [18]. However, since coalitions problems allow to state succinctly that global and coalition clauses hold in all states of a model, the satisfiability problem for coalition problems in unit $\mathsf{DSNF}_{\mathsf{CL}}$, $\mathsf{DSNF}_{\mathsf{CL}}$, and $\mathsf{DSNF}_{\mathsf{VCL}}$ is EXPTIME-hard [16].

As we have argued above, the number of distinct initial, universal and coalition problems that can be formed over the set of propositions $\Pi_{\mathcal{C}}$ occurring in a coalition problem $\mathcal{C}$ in $\mathsf{DSNF}_{\mathsf{VCL}}$ and set of numbers $I_{\mathcal{C}}$ occurring in coalition vectors in $\mathcal{C}$, is exponential in the size of $\Pi_{\mathcal{C}}$ and $I_{\mathcal{C}}$, and therefore also exponential in the size of $\mathcal{C}$. Each inference step by $\mathsf{RES}^{\succ}_{\mathsf{CL}}$ requires polynomial time in the size of the clause set. Overall, this implies a decision procedure based on $\mathsf{RES}^{\succ}_{\mathsf{CL}}$ is in EXPTIME.

Theorem 7. *The complexity of a $\mathsf{RES}_{\mathsf{CL}}$ based decision procedure for the satisfiability problem in CL is EXPTIME.*

5 Implementation and Evaluation

CLProver++ [8] is a C++ implementation of the resolution based calculus $\mathsf{RES}^{\succ}_{\mathsf{CL}}$ described in Section 4. CLProver++ also implements unit propagation, pure literal elimination, tautology elimination, forward subsumption and backward subsumption. *Feature vector indexing*, a non-perfect indexing method first introduced by Schultz in [20], is used to store coalition problems and to retrieve a superset of candidates for subsumption or resolution efficiently. When selecting the next clause as main premise for resolution inferences, CLProver++ will choose the smallest clause in a coalition problem and will prefer universal clauses over initial clauses over coalition clauses of the same length.

To evaluate the performance of CLProver++ we will compare it with CLProver and TATL (September 2014 version). CLProver [17] is a prototype implementation in SWI-Prolog of the calculus $\mathsf{RES}_{\mathsf{CL}}$. It also implements forward subsumption but uses no heuristics to guide the search for a refutation. TATL [6] is an implementation in OCaml of the two-phase tableau calculus by Goranko and Shkatov for ATL. Note that the construction phase for CL is the same as for ATL (there is no additional overhead) and the additional elimination rule required for ATL in the elimination phase does not need to be engaged for CL.

While a limited number of coalition logic formulae can be found in the literature [18], they prove to be insufficiently challenging to evaluate the performance of decision procedures for coalition logic.

We therefore use randomly generated CL formulae for that purpose, in particular, we have devised two classes of benchmark formulae[5], $\mathfrak{B}_1$ and $\mathfrak{B}_2$. $\mathfrak{B}_1$ consists of randomly generated formulae without any particular structure containing at most 5 propositional symbols and at most 2 agents. A feature of these

[5] Available at `http://cgi.csc.liv.ac.uk/~ullrich/CLProver++/`

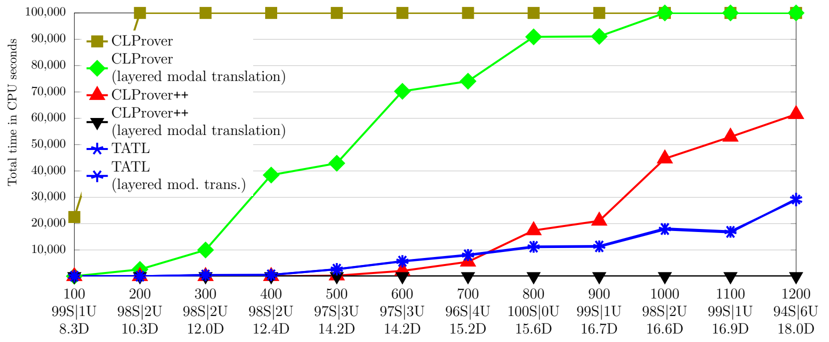

Figure 3. Performance of CLProver, CLProver++, and TATL on $\mathfrak{B}_1$.

formulae is their relatively high modal depth. For lengths[6] L between 100 and 1200, in steps of 100, we have generated 100 such formulae. $\mathfrak{B}_2$ consists of randomly generated formulae in conjunctive normal form where half the conjuncts are unary disjunctions and half are binary disjunctions, and each disjunct is of the form $[\mathcal{A}](l_1 \vee l_2)$ or $\neg[\mathcal{A}](l_1 \vee l_2)$, written $(\neg)[\mathcal{A}](l_1 \vee l_2)$, with l_1 and l_2 being random propositional literals over 5 propositional symbols, $\mathcal{A}$ is a random subset of $\{1, 2\}$, and the probability of a disjunct or propositional literal being negative is 0.5:

$$\begin{array}{rl} (\neg)[\mathcal{A}_1^1](l_1^1 \vee l_2^1) \wedge & ((\neg)[\mathcal{A}_1^2](l_1^2 \vee l_2^2) \vee (\neg)[\mathcal{A}_2^2](l_3^2 \vee l_4^2)) \\ \wedge \ldots \wedge (\neg)[\mathcal{A}_1^{L-1}](l_1^{L-1} \vee l_2^{L-1}) \wedge & ((\neg)[\mathcal{A}_1^L](l_1^L \vee l_2^L) \vee (\neg)[\mathcal{A}_2^L](l_3^L \vee l_4^L)). \end{array}$$

For numbers L of conjuncts between 2 and 24, in steps of 2, we have again generated 100 such formulae.

For both classes, the number of agents and propositional variables were chosen to allow all provers to solve most of the formulae involved while also not being trivial for all provers. The particular structure of the formulae in $\mathfrak{B}_2$ was chosen for the same reasons.

Figures 3 and 4 show the performance of the three provers on $\mathfrak{B}_1$ and $\mathfrak{B}_2$, respectively, measured on PCs with Intel i7-2600 CPU @ 3.40GHz and 16GB main memory. The labels on the x-axis take the form 's nS|mU' where s is the length or number of conjuncts of the formulae (their *size*), n is the number of satisfiable formulae and m is the number of unsatisfiable formulae among the 100 formulae of size s. For $\mathfrak{B}_1$ we also indicate by 'kD' the average modal depth k of formulae. For each formula φ we have measured the time it took each of the provers to solve φ, stopping the execution of a prover after 1000 CPU seconds The time to transform a formula into a coalition problem is not included, but is negligible. The figures show the total number of CPU seconds it took each prover to attempt or solve the 100 formulae of size s.

We can see that almost all formulae in $\mathfrak{B}_1$ are satisfiable, independent of the value of L. We can also see that CLProver++ performs better than TATL for

[6] The length of a CL formula φ is the number of occurrences of propositional symbols, propositional logical operators, and modal operators in φ.

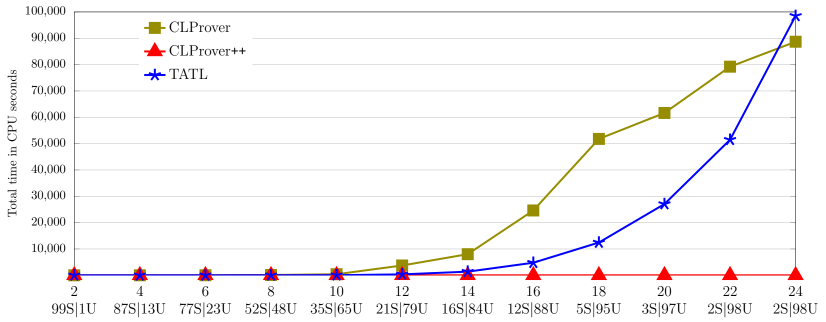

Figure 4. Performance of CLProver, CLProver++, and TATL on $\mathfrak{B}_2$.

$L < 800$, but falls behind for values of $L \geq 800$. This is because formulae in $\mathfrak{B}_1$ contain numerous occurrences of the five propositional symbols at various different modal depths. In Coalition Logic, just as in basic modal logic, the truth values of propositional symbols occurring at different modal depths are independent of each other, something that TATL takes advantage of. In contrast, the transformation to $\mathsf{DSNF}_{\mathsf{CL}}$ 'flattens' formulae, leading to unnecessary inferences by CLProver++ and CLProver. It is possible to pre-process CL formulae using the 'layered modal translation' technique by Areces et al. [2, Definition 4.1] which replaces a propositional symbol p occurring at modal depth n by a new, unique propositional symbol p_n. If we do so for $\mathfrak{B}_1$ formulae before transformation to $\mathsf{DSNF}_{\mathsf{CL}}$, the performance of CLProver++ improves dramatically, to near zero time, as indicated by the data for 'CLProver++ (layered modal translation)' in Figure 3. CLProver also shows a considerable improvement while, as one would expect, TATL shows no improvement, as indicated by the data for 'CLProver (layered modal translation)' and 'TATL (layered mod. trans.)', respectively.

Regarding $\mathfrak{B}_2$, we see the expected decline in the proportion of satisfiable formulae as the number L of conjuncts in the formulae increases. This class proves to be very easy for CLProver++ while CLProver and TATL take increasingly more time to solve formulae as L increases and the number of formulae that can be solved within the time limit drops sharply. The runtime of TATL appears to be dominated by the time required for the construction phase and pre-state deletion phase, meaning satisfiable formulae are on average not solved faster than unsatisfiable formulae of the same size. On more challenging formulae than those in $\mathfrak{B}_1$ and $\mathfrak{B}_2$, CLProver++ is, as expected, considerably faster on unsatisfiable formulae than on satisfiable formulae. In contrast, CLProver has no heuristics to guide its search for a refutation, so, just as for TATL, unsatisfiable formulae are not solved significantly faster than satisfiable formulae of the same size.

It is clear from Figures 3 and 4 that CLProver++ performs much better than CLProver. To understand the contribution made by the ordering refinement, we look at the number of inferences performed by each of the two provers on formulae from $\mathfrak{B}_2$ in unit $\mathsf{DSNF}_{\mathsf{CL}}$ solved by both provers when using the same function to select the next clause for resolution inferences. The difference in the

		Inferences by					Inferences by		
Size	Solved	CLProver	CLProver++	Ratio	Size	Solved	CLProver	CLProver++	Ratio
2	100	1335	176	7.6	14	98	2519336	8766	287.4
4	100	9475	969	9.8	16	93	5098117	10300	495.0
6	100	39220	2046	19.2	18	64	4299430	9145	470.1
8	100	169707	3357	50.6	20	29	1743043	4788	364.0
10	100	394952	4824	81.9	22	30	1879064	6173	304.4
12	100	1835089	6746	272.0	24	16	1008285	3749	269.0

Figure 5. Total number of inferences by CLProver and CLProver++ on $\mathfrak{B}_2$.

number of inferences performed by CLProver and CLProver++ is independent of the differences between $\mathsf{DSNF}_{\mathsf{CL}}$ and $\mathsf{DSNF}_{\mathsf{VCL}}$, the different programming languages used for the implementation of the provers, the different data structures they use to store clauses, and, as far as this is possible, their heuristics for selecting clauses. Figure 5 shows that, on average, CLProver++ performs 220 times fewer inferences than CLProver.

6 Conclusion and Future Work

We have described a calculus $\mathsf{RES}^{\succ}_{\mathsf{CL}}$ based on ordered resolution for Coalition Logic and sketched proofs of its soundness and completeness. We have also shown that any derivation by $\mathsf{RES}^{\succ}_{\mathsf{CL}}$ terminates. The prover CLProver++ provides an implementation of $\mathsf{RES}^{\succ}_{\mathsf{CL}}$. Our evaluation of CLProver++ indicates that the ordering refinement improves performance by several orders of magnitude compared to unrefined resolution as implemented in CLProver. Our evaluation also shows that similar improvements can be gained by optimising the normal form transformation that is used to obtain coalition problems from CL formulae.

Our work on Coalition Logic is a first step towards the development of resolution calculi for more expressive logics for reasoning about the strategic abilities of coalitions of agents. A wide variety of such logics can be found in the literature, starting with Alternating-Time Temporal Logic ATL. The notion of coalition vectors that we have introduced in this paper are closely related to the notion of k-actions in Coalition Action Logic [5] and to the notion of commitment functions in ATLES [22]. We believe that the combination of the techniques developed in this paper with the techniques for temporal logics with eventualities provide a good basis for the development of effective calculi for logics such as ATL, Coalition Action Logic and ATLES.

References

1. Alur, R., Henzinger, T.A., Kupferman, O.: Alternating-time temporal logic. J. ACM 49(5), 672–713 (2002)
2. Areces, C., Gennari, R., Heguiabehere, J., de Rijke, M.: Tree-Based heuristic in modal theorem proving. In: Proc. ECAI2000. IOS Press (2000)

3. Areces, C., Gorín, D.: Resolution with order and selection for hybrid logics. J. of Automated Reasoning 46, 1–42 (2011), 10.1007/s10817-010-9167-0
4. Bachmair, L., Ganzinger, H.: Resolution theorem proving. In: Robinson, A., Voronkov, A. (eds.) Handbook of Automated Reasoning, pp. 19–99. Elsevier (2001)
5. Borgo, S.: Coalitions in action logic. In: Proc. IJCAI '07. pp. 1822–1827 (2007)
6. David, A.: TATL: Implementation of ATL tableau-based decision procedure. In: Proc. TABLEAUX 2013. LNCS, vol. 8123, pp. 97–103. Springer (2013)
7. van Drimmelen, G.: Satisfiability in alternating-time temporal logic. In: Proc. LICS '03. pp. 208–207. IEEE (2003)
8. Gainer, P., Hustadt, U., Dixon, C.: CLProver++ (2015), http://cgi.csc.liv.ac.uk/~ullrich/CLProver++/
9. Goranko, V.: Coalition games and alternating temporal logics. In: Proc. TARK '01. pp. 259–272. Morgan Kaufmann (2001)
10. Goranko, V., van Drimmelen, G.: Complete axiomatization and decidability of alternating-time temporal logic. Theor. Comput. Sci. 353(1), 93–117 (2006)
11. Goranko, V., Shkatov, D.: Tableau-based decision procedures for logics of strategic ability in multiagent systems. ACM Trans. Comput. Log. 11(1), 1–51 (2009)
12. Hansen, H.H.: Tableau games for coalition logic and alternating-time temporal logic – theory and implementation. Master's thesis, University of Amsterdam, The Netherlands (Oct 2004)
13. Horrocks, I., Hustadt, U., Sattler, U., Schmidt, R.A.: Computational modal logic. In: Blackburn, P., van Benthem, J., Wolter, F. (eds.) Handbook of Modal Logic, pp. 181–245. Elsevier (2006)
14. Hustadt, U., Konev, B.: TRP++: A temporal resolution prover. In: Baaz, M., Makowsky, J., Voronkov, A. (eds.) Collegium Logicum, pp. 65–79. Kurt Gödel Society (2004)
15. Kovács, L., Voronkov, A.: First-order theorem proving and Vampire. In: Proc. CAV 2013. LNCS, vol. 8044, pp. 1–35. Springer (2013)
16. Nalon, C., Zhang, L., Dixon, C., Hustadt, U.: A resolution-based calculus for coalition logic. J. Logic Comput. 24(4), 883–917 (2014)
17. Nalon, C., Zhang, L., Dixon, C., Hustadt, U.: A resolution prover for coalition logic. In: Proc. SR2014. Electron. Proc. Theor. Comput. Sci., vol. 146, pp. 65–73 (2014)
18. Pauly, M.: Logic for Social Software. Ph.D. thesis, University of Amsterdam, The Netherlands (2001)
19. Plaisted, D.A., Greenbaum, S.: A structure-preserving clause form translation. J. Symb. Comput. 2, 293–304 (1986)
20. Schulz, S.: Simple and efficient clause subsumption with feature vector indexing. In: Automated Reasoning and Mathematics: Essays in Memory of William W. McCune, LNCS, vol. 7788, pp. 45–67. Springer (2013)
21. Schulz, S.: System description: E 1.8. In: Proc. LPAR-19. LNCS, vol. 8312, pp. 735–743. Springer (2013)
22. Walther, D., van der Hoek, W., Wooldridge, M.: Alternating-time temporal logic with explicit strategies. In: Proc. TARK '07. pp. 269–278. ACM (2007)
23. Walther, D., Lutz, C., Wolter, F., Wooldridge, M.: ATL satisfiability is indeed ExpTime-complete. J. Logic Comput. 16(6), 765–787 (2006)
24. Weidenbach, C., Dimova, D., Fietzke, A., Kumar, R., Suda, M., Wischnewski, P.: SPASS version 3.5. In: Schmidt, R.A. (ed.) Proc. CADE-22. LNCS, vol. 5663, pp. 140–145. Springer (2009)
25. Zhang, L., Hustadt, U., Dixon, C.: A resolution calculus for branching-time temporal logic CTL. ACM Trans. Comput. Log. 15(1), 10:1–38 (2014)