CLProver++: An Ordered Resolution Prover for Coalition Logic

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

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

Abstract: We present CLProver++, a theorem prover for Coalition Logic a non-normal modal logic for reasoning about cooperative agency. CLProver++ is based on recent work on an ordered resolution calculus for Coalition Logic. We provide an overview of this calculus, give some details of the implementation of CLProver++ and present an evaluation of the performance of CLProver++.

1 Introduction

Coalition Logic CL was introduced by Pauly [5] as a logic for reasoning about what groups of agents can bring about by collective action. CL is a propositional modal logic over a countably infinite set Π of propositional symbols and a non-empty, finite set $\Sigma \subset \mathbb{N}$ of agents with modal operators of the form $[\mathcal{A}]$, where $\mathcal{A} \subseteq \Sigma$. 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*. Formally, the semantics of CL formulae is given by *Concurrent Game Models* (CGMs), see [3]. The satisfiability problem of CL is PSPACE-complete [5]. Various decision procedures for Coalition Logic exist including tableaux and unrefined resolution calculi.

In the following we present a new decision procedure for CL based on ordered resolution, briefly describe its implementation and present its evaluation.

2 Ordered Resolution for CL

Our ordered resolution calculus does not operate on CL formulae, but on formulae of *Vector Coalition Logic* in a clausal normal form.

Let $|\Sigma| = k$. A *coalition vector* $\vec{c}$ is a k-tuple such that for every a, $1 \leq a \leq k$, $\vec{c}[a]$ 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 $\vec{c}[a] = \vec{c}[a']$.

The set $\mathsf{WFF}_{\mathsf{VCL}}$ of *Vector Coalition Logic (*VCL*) formulae* is inductively defined as follows: (i) if p is a propositional symbol in Π, then p and $\neg p$ are VCL formulae; (ii) if φ is a propositional formula and ψ is a VCL formula, then $(\varphi \to \psi)$ is a VCL formula; (iii) 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 (iv) if $\vec{c}$ is a coalition vector and φ is a VCL formula, then so is $\vec{c}\varphi$. The semantics of $\mathsf{WFF}_{\mathsf{VCL}}$ formulae is given by Concurrent Game Models extended with *choice functions* ($\mathrm{CGM}_{\mathrm{CF}}$) that give meaning to coalition vectors. Intuitively, a coalition vector represents the choices made by each agent. Each number represents a choice function that selects an agent's move (action) depending on the current world and, possibly, the moves of other agents.

A *coalition problem in $\mathsf{DSNF}_{\mathsf{VCL}}$* is a tuple $(\mathcal{I},\mathcal{U},\mathcal{N})$ such that $\mathcal{I}$ is a set of *initial clauses*, and $\mathcal{U}$ is a set of *global clauses*, are finite sets of propositional clauses $\bigvee_{j=1}^{n} l_j$, 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.

Intuitively, initial clauses are true at one distinguished world in a $\mathrm{CGM}_{\mathrm{CF}}$ while global and coalition clauses are true at every world in a $\mathrm{CGM}_{\mathrm{CF}}$, the later imposing a constraint $\bigvee_{j=1}^{n} l_j$ on the worlds that a coalition can 'reach' from a world w by its actions, provided the condition $\bigwedge_{i=1}^{m} l'_i$ is satisfied at the world w.

There are two more ingredients to our calculus that we need to introduce, the notion of the merge of two coalition vectors and the notion of atom orderings.

Let $\vec{c}_1$ and $\vec{c}_2$ be two coalition vectors of length k. 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, $1 \leq i \leq k$, 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 of both $\vec{c}_1$ and $\vec{c}_2$. A coalition vector $\vec{c}_3$ is a *merge* of $\vec{c}_1$ and $\vec{c}_2$, denoted $\vec{c}_1 \downarrow \vec{c}_2$, 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*.

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 $q \in \Pi$ 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, $l' \not\succ l$.

The ordered resolution calculus $\mathsf{RES}^{\succ}_{\mathsf{CL}}$ is then given by the rules shown in Figure 1.

Theorem 1 *Let φ be a* CL *formula. Then there is a coalition problem $\mathcal{C}$ in $\mathsf{DSNF}_{\mathsf{VCL}}$ that is satisfiable if and only*

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

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

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

$$\textbf{VRES2}\qquad \begin{array}{rcll} & & C\lor l & \in\mathcal{U}\\ Q & \to & \vec{c}(D\lor\neg l) & \in\mathcal{N}\\ \hline Q & \to & \vec{c}(C\lor D) & \in\mathcal{N}\end{array}$$

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

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 1: Resolution Calculus $\mathsf{RES}^{\succ}_{\mathsf{CL}}$

if φ is satisfiable. Furthermore, any derivation by $\mathsf{RES}^{\succ}_{\mathsf{CL}}$ from $\mathcal{C}$ terminates and φ is unsatisfiable if and only if there is a refutation of $\mathcal{C}$ by $\mathsf{RES}^{\succ}_{\mathsf{CL}}$.

3 CLProver++

CLProver++ [2] is a C++ implementation of the resolution based calculus $\mathsf{RES}^{\succ}_{\mathsf{CL}}$ described in Section 2. CLProver++ also implements unit propagation, pure literal elimination, forward subsumption and backward subsumption. Clauses in a coalition problem are split into a set Wo of worked-off clauses and set Us of usable clauses. The main loop of the prover heuristically selects a clause G from Us, moves it to Wo and performs all inferences between G and clauses in Wo. The set New of newly derived clauses is subject to forward subsumption and the remaining clauses in New may optionally be used to backward subsume clauses in Us and Wo. *Feature vector indexing* [6], a non-perfect indexing method, is used to store Us and Wo, and to retrieve a superset of candidates for subsumption or resolution efficiently.

To evaluate the performance of CLProver++ we have compared it with CLProver and TATL (September 2014 version). CLProver [4] 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 [1] is an implementation in OCaml of the two-phase tableau calculus by Goranko and Shkatov for ATL [3], that can also be used to decide the satisfiability of CL formulae.

We have used two classes $\mathfrak{B}_1$ and $\mathfrak{B}_2$ of randomly generated CL formulae for the evaluation that are available

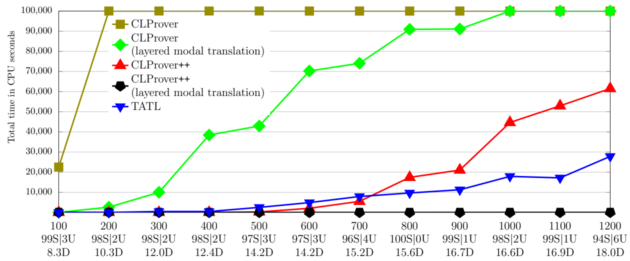

Figure 2: Performance on $\mathfrak{B}_1$.

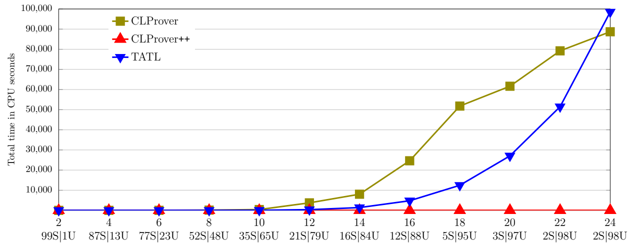

Figure 3: Performance on $\mathfrak{B}_2$.

from the CLProver++ website [2]. $\mathfrak{B}_1$ consists of twelve sets S_i, $1\le i\le 12$, of 100 formulae each, with each formula in S_i having length $100\times i$. $\mathfrak{B}_2$ consists of 12 sets S_i, $1\le i\le 12$ of 100 formulae in conjunctive normal form with i conjuncts of the form $(\neg)[\mathcal{A}^1_1](l^1_1\lor l^1_2)\land((\neg)[\mathcal{A}^2_1](l^2_1\lor l^2_2)\lor(\neg)[\mathcal{A}^2_2](l^2_3\lor l^2_4))$ with elements of each conjunct generated randomly.

Figures 2 and 3 show the total runtime of each of the provers on each of the sets in $\mathfrak{B}_1$ and $\mathfrak{B}_2$, respectively. Execution of a prover on a formula was stopped after 1000 CPU seconds. The time to transform a formulae into a coalition problem is not included, but is negligible. Overall, CLProver++ outperforms all other systems by a large margin.

References

[1] A. David. TATL: Implementation of ATL tableau-based decision procedure. In *Proc. TABLEAUX 2013*, *LNCS* 8123:97–103. Springer, 2013.

[2] P. Gainer, U. Hustadt, C. Dixon. CLProver++, 2015. http://cgi.csc.liv.ac.uk/~ullrich/CLProver++/.

[3] V. Goranko and D. Shkatov. Tableau-based decision procedures for logics of strategic ability in multiagent systems. *ACM Trans. Comput. Log.*, 11(1):1–51, 2009.

[4] C. Nalon, L. Zhang, C. Dixon, and U. Hustadt. A resolution prover for coalition logic. In *Proc. SR2014*, *Electron. Proc. Theor. Comput. Sci.*, 146:65–73, 2014.

[5] M. Pauly. *Logic for Social Software*. PhD thesis, University of Amsterdam, The Netherlands, 2001.

[6] S. Schulz. Simple and efficient clause subsumption with feature vector indexing. In *Automated Reasoning and Mathematics: Essays in Memory of William W. McCune*, *LNCS* 7788:45–67. Springer, 2013.