Description Logic for Coalitions

İnanç Seylan
Faculty of Computer Science
Free University of Bozen-Bolzano, Italy
seylan@inf.unibz.it

Wojciech Jamroga
Department of Informatics
Clausthal University of Technology, Germany
wjamroga@in.tu-clausthal.de

ABSTRACT

Coalition Logic (**CL**) is one of the most important formalisms for specification and verification of game-like multi-agent systems. Several extensions of the logic have been studied in the literature. These extensions are usually fusions (independent joins) of **CL** with other modal logics (e.g., temporal, epistemic, dynamic, etc.), and they are generally propositional. In this paper, we propose a game description logic called $\mathbf{CL}_{\mathcal{ALC}}$ which is based on a product of Coalition Logic with the description logic $\mathcal{ALC}$. The new logic allows one to reason about agents' ability to influence first-order structures. We show that the satisfiability problem for $\mathbf{CL}_{\mathcal{ALC}}$ is decidable; we prove this by giving a goal-directed decision procedure for the problem.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent Systems*; I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods—*Modal logic*

General Terms

Theory, Algorithms

Keywords

Strategic logics, description logics, satisfiability, tableaux

1. INTRODUCTION

Coalition logic **CL** [10, 11] formalizes the ability of groups of agents to achieve certain outcomes in strategic games. The central operator of **CL** is $[A]$, with $[A]\varphi$ meaning that group of agents A has a strategy to achieve an outcome state where φ holds. The logic has important applications in the specification and verification of game-like scenarios, social choice mechanisms (e.g., design of voting protocols), etc. The latter group of applications is closely related to the satisfiability problem for **CL**: showing that a **CL** specification φ is satisfiable amounts in most cases to construction of a model (mechanism, protocol) that satisfies φ.

Cite as: Description Logic for Coalitions, Inanç Seylan, Wojciech Jamroga, *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009),* Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary, pp. 425–432

Several extensions of **CL** have been proposed (see [15] and the references therein). The extensions focus on the representation of actions, preferences, incomplete information, origins of power, social laws, and quantification over agents and coalitions. A common feature of the resulting logics is that the underlying language is propositional. In consequence, we can use them only to address simple properties of games and states, but not properties of individual entities that are involved in the game (like people, places, messages, communication channels etc.). For example, the only way to say that agent a can make all the sent messages reach their recipients is to create a proposition that labels all the states where this is the case (e.g., $\mathsf{allSentReceived}$), and then write $[a]\mathsf{allSentReceived}$. Of course, this method of specification is neither elegant nor flexible, and becomes impractical for all but simplest scenarios.

Description Logics (DLs) are logical formalisms for representing the knowledge of an application domain in a structured way [3]. More precisely, DLs allow to describe classes, assign individuals to these classes, and define binary relations on individuals. For instance, we can use DL terms Sent and $\mathsf{Received}$ for the classes of sent and received messages. Then, the DL formula $\mathsf{Sent} \sqsubseteq \mathsf{Received}$ says that every sent message is received too. Note that the dual statement: "*some* sent messages have been received" can be expressed by formula $\neg(\mathsf{Sent} \sqcap \mathsf{Received} = \bot)$.

Description Logics are important because they are *decidable* fragments of first order logic. Our combination of a DL with coalition logic brings first-order perspective to reasoning about coalitional abilities, while keeping it still decidable. Furthermore, DLs have well developed practical decision procedures. Last but not least, they comprise the formal basis of the Semantic Web ontology languages [6]. The Semantic Web is built on the vision of giving explicit meaning to information, making it easier for software agents to automatically process and integrate information available on the Web. Combining agent logics with DLs enables reasoning about how (and by whom) the information can be manipulated, which is potentially interesting for both the agents community and the Semantic Web community.

The logic that we propose in this paper is a product style combination of the description logic $\mathcal{ALC}$ with coalition logic, that is interpreted over constant domain models. The resulting multi-modal logic called $\mathbf{CL}_{\mathcal{ALC}}$ allows for application of modal operators to both formulas and concepts. Also, concept names and role names are interpreted locally. For example, the $\mathbf{CL}_{\mathcal{ALC}}$ formula $[a](\mathsf{Sent} \sqsubseteq \mathsf{Received})$ can be used to express that agent a can make the sent mes-

sages be received. Another meaningful $\mathbf{CL}_{\mathcal{ALC}}$ specification $\mathsf{Sent} \sqsubseteq [a]\mathsf{Received}$ says that, for every message that has been *already* sent (prior to a's involvement), a can guarantee its reception. We emphasize that the combination of logics, studied here, is not trivial. This is because $\mathbf{CL}_{\mathcal{ALC}}$ is closely related to the Cartesian product $\mathbf{CL} \times \mathbf{S5}$ (cf. [14, 8] for a more detailed discussion).

Combinations of various modal logics with DLs have been studied extensively (see [2] and the references therein). In [4], a tableau algorithm for a fusion style extension of $\mathcal{ALC}$ with belief and intention modalities was developed. Another relevant paper is [12] which studied the simple logic $\mathbf{M}_{\mathcal{ALC}}$ obtained by combining $\mathcal{ALC}$ with monotonic modal operators that could be applied to formulas and concepts. In this paper, we show that the satisfiability problem of $\mathbf{CL}_{\mathcal{ALC}}$ is decidable by giving a tableau decision procedure for it. The algorithm presented in this paper is developed incrementally (similarly to [5]): we start with the decision procedure for $\mathbf{M}_{\mathcal{ALC}}$ from [12], and extend it to handle $\mathbf{CL}_{\mathcal{ALC}}$.

The paper is organized as follows. First the logic $\mathbf{CL}_{\mathcal{ALC}}$ is introduced. Then we define structures that are equivalent to $\mathbf{CL}_{\mathcal{ALC}}$ models. These structures enable us to reason with $\mathbf{CL}_{\mathcal{ALC}}$ formulas in a more convenient way. Next, we present our tableau based decision procedure for the satisfiability of $\mathbf{CL}_{\mathcal{ALC}}$ formulas and show its correctness. Finally, we conclude the work. Proofs of some lemmas are left out due to space limits; they can be found in the technical report [13].

2. COALITIONAL DESCRIPTION LOGIC

In this section, we introduce our logic $\mathbf{CL}_{\mathcal{ALC}}$. The logic combines the first-order perspective of the basic description logic $\mathcal{ALC}$ with strategic modalities of Coalition Logic. On the syntactical level, $\mathcal{ALC}$ contributes terms for individuals and their classes (i.e., concepts), while $\mathbf{CL}$ adds operators for reasoning about outcome of strategies and dynamics of concepts. On the semantic level, models of $\mathbf{CL}$ (which can be roughly understood as strategic games played successively one after another) are enriched with concept structures that can evolve over time.

2.1 Syntax

DEFINITION 1. *Let $\mathbb{A}\mathrm{gt}$ be a finite non-empty set of agents, and let N_C and N_R be countably infinite sets of concept names and role names, respectively. Modal operators $[A]$ and $\langle A\rangle$ are associated with every coalition $A \subseteq \mathbb{A}\mathrm{gt}$. $\wedge$, $\vee$, and $\neg$ represent standard logical connectives. Every concept name in N_C as well as $\top$ (top concept) and $\bot$ (bottom concept) are concepts. Let C and D be concepts, R a role name in N_R, and $A \subseteq \mathbb{A}\mathrm{gt}$. Then $\neg C$, $C \sqcap D$, $C \sqcup D$, $\forall R.C$, $\exists R.C$, $[A]C$, and $\langle A\rangle C$ are concepts. $C \sqsubseteq D$ and $C = D$ are atomic formulas. If φ and ψ are formulas then so are $\neg\varphi$, $\varphi \wedge \psi$, $\varphi \vee \psi$, $[A]\varphi$, and $\langle A\rangle\varphi$.*

That is, atomic formulas compare concepts, and modal formulas can refer to coalitional ability to enforce a particular relationship between concepts. For instance, formula $[\{jumbo, clown\}](\mathsf{Person} \sqsubseteq \mathsf{Happy})$ says that Jumbo and the clown can make every person happy, and $\langle\{clown\}\rangle\neg(\mathsf{Person} \sqcap \mathsf{Sad} = \bot)$ states that the clown cannot prevent (on his own) some persons from being sad. Concepts are either primitive or built from simpler ones by use of constructors $\neg, \sqcap, \sqcup$, etc. We also add "strategic" concept constructors: $[\{clown\}]\mathsf{Happy}$ reads as "the set of individuals that can be turned happy by the clown", and $\langle\{jumbo\}\rangle\mathsf{Sad}$ as "those that Jumbo cannot prevent from becoming sad".

While writing modal operators, we will omit set braces from coalitions; for instance, we will write $[1, 2]$ instead of $[\{1, 2\}]$. Note that named individuals are not allowed in the DL part of our language for ease of presentation. The interested reader is referred, e.g., to [8] to see how named individuals can be dealt with.

2.2 Semantics

The semantics of $\mathbf{CL}_{\mathcal{ALC}}$ joins the first-order interpretation of concepts from $\mathcal{ALC}$ with the possible world semantics of $\mathbf{CL}$ operators. The interpretation of a concept can evolve over time as a result of strategic choices of agents. However, we assume for simplicity that the domain of interpretation does not change from state to state.

DEFINITION 2. *A coalition model for $\mathbf{CL}_{\mathcal{ALC}}$ is a triple of the form $\mathfrak{M} = \langle W, E, \mathcal{I}\rangle$, where W is a non-empty set of states, E is a mapping that associates a playable[1] effectivity function $E_w : 2^{\mathbb{A}\mathrm{gt}} \to 2^{2^W}$ with each $w \in W$, and $\mathcal{I}$ is a function associating with each $w \in W$ an $\mathcal{ALC}$ interpretation $\mathcal{I}(w) = \left\langle \Delta^{\mathcal{I}(w)}, \cdot^{\mathcal{I}(w)} \right\rangle$. An element V of $E_w(A)$ i.e., a subset of W, is called an outcome. $\Delta^{\mathcal{I}(w)}$ is a non-empty set called the domain of state w, and $\cdot^{\mathcal{I}(w)}$ maps each concept name C to a subset $C^{\mathcal{I}(w)}$ of $\Delta^{\mathcal{I}(w)}$ and each role name R to a binary relation $R^{\mathcal{I}(w)}$ on $\Delta^{\mathcal{I}(w)}$. For any $w, v \in W$, we have $\Delta^{\mathcal{I}(w)} = \Delta^{\mathcal{I}(v)}$ (constant domain assumption).*

DEFINITION 3 ([10]). *An effectivity function E_w is* playable *iff it satisfies the following conditions:*

(C1) E_w is serial*: $\emptyset \notin E_w(A)$ for all coalitions A.*

*(C2) E_w is W-*complete*: $W \in E_w(A)$. for all coalitions A.*

*(C3) E_w is $\mathbb{A}\mathrm{gt}$-*maximal*: for all V, if $\overline{V} \notin E_w(\emptyset)$ then $V \in E_w(\mathbb{A}\mathrm{gt})$.*

(C4) E_w is outcome-monotonic*: for all $V \subseteq U \subseteq W$ and for all A, if $V \in E_w(A)$ then $U \in E_w(A)$.*

(C5) E_w is superadditive*: for all V, U, A_1, and A_2 such that $A_1 \cap A_2 = \emptyset$, if $V \in E_w(A_1)$ and $U \in E_w(A_2)$ then $V \cap U \in E_w(A_1 \cup A_2)$.*

EXAMPLE 1. *Consider a system that consists of a sender s, a receiver r, and a communication channel c. The sender can either do nothing (action nop), or send message m_1 (action $send_1$) or m_2 (action $send_2$). The channel can either transmit the message to the receiver (action t) or ignore it (action i). We assume that the receiver receives incoming messages automatically. Thus, he does not execute any relevant actions, and it is enough to include only actions of the sender and the channel in the model. The action structure of the system is depicted in Figure 1 (note: only outgoing transitions from states w_0 and w_1 are shown).*

The domain of interpretation contains only messages, i.e., $\Delta^{\mathcal{I}(w)} = \{m_1, m_2\}$ for all w. There are two primitive concepts: Sent *and* $\mathsf{Received}$ *that are used to register the messages that have been sent (resp. received) until the current moment. The interpretation of* Sent *($\mathsf{Sent}^{\mathcal{I}(w)}$) is denoted by S in the graph;* $\mathsf{Received}^{\mathcal{I}(w)}$ *is referred to with R.*

[1] See below for the definition of playability.

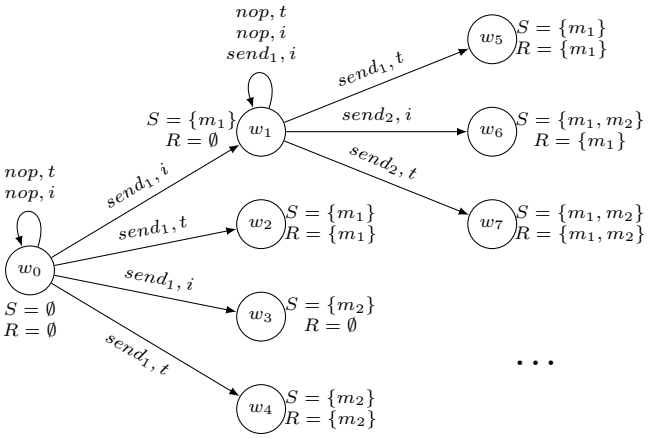

Figure 1: Sending messages through a fitful channel

The interpretation $\mathcal{I}(w)$ defines the semantics of primitive concepts in state w. We extend it to concept descriptions in the standard DL fashion:

$$\begin{aligned}
\top^{\mathcal{I}(w)} &= \Delta^{\mathcal{I}(w)},\\
\bot^{\mathcal{I}(w)} &= \emptyset,\\
(\neg C)^{\mathcal{I}(w)} &= \Delta^{\mathcal{I}(w)} \setminus C^{\mathcal{I}(w)},\\
(C \sqcap D)^{\mathcal{I}(w)} &= C^{\mathcal{I}(w)} \cap D^{\mathcal{I}(w)},\\
(C \sqcup D)^{\mathcal{I}(w)} &= C^{\mathcal{I}(w)} \cup D^{\mathcal{I}(w)},\\
(\forall R.C)^{\mathcal{I}(w)} &= \{\delta \in \Delta^{\mathcal{I}(w)} \mid \forall \delta' \, (\langle \delta, \delta' \rangle \in R^{\mathcal{I}(w)} \rightarrow \delta' \in C^{\mathcal{I}(w)})\},\\
(\exists R.C)^{\mathcal{I}(w)} &= \{\delta \in \Delta^{\mathcal{I}(w)} \mid \exists \delta' \, (\langle \delta, \delta' \rangle \in R^{\mathcal{I}(w)} \wedge \delta' \in C^{\mathcal{I}(w)})\}.
\end{aligned}$$

Moreover, we add the following definitions:

$$\begin{aligned}
([A]C)^{\mathcal{I}(w)} &= \{\delta \in \Delta^{\mathcal{I}(w)} \mid \|C\|^{\mathfrak{M}}_{\delta} \in E_w(A)\},\\
(\langle A \rangle C)^{\mathcal{I}(w)} &= \{\delta \in \Delta^{\mathcal{I}(w)} \mid W \setminus \|C\|^{\mathfrak{M}}_{\delta} \notin E_w(A)\},
\end{aligned}$$

where $\|C\|^{\mathfrak{M}}_{\delta} = \{w \in W \mid \delta \in C^{\mathcal{I}(w)}\}$ is the set of states that δ belongs to concept C.

Definition 4. *The satisfaction relation $\models$ for $\mathbf{CL}_{\mathcal{ALC}}$ is defined as follows:*

$$\begin{aligned}
&\mathfrak{M}, w \models C \sqsubseteq D \quad \text{iff} \quad C^{\mathcal{I}(w)} \subseteq D^{\mathcal{I}(w)},\\
&\mathfrak{M}, w \models C = D \quad \text{iff} \quad C^{\mathcal{I}(w)} = D^{\mathcal{I}(w)},\\
&\mathfrak{M}, w \models \neg\varphi \quad \text{iff} \quad \mathfrak{M}, w \not\models \varphi,\\
&\mathfrak{M}, w \models \varphi \wedge \psi \quad \text{iff} \quad \mathfrak{M}, w \models \varphi \text{ and } \mathfrak{M}, w \models \psi,\\
&\mathfrak{M}, w \models \varphi \vee \psi \quad \text{iff} \quad \mathfrak{M}, w \models \varphi \text{ or } \mathfrak{M}, w \models \psi,\\
&\mathfrak{M}, w \models [A]\varphi \quad \text{iff} \quad \|\varphi\|^{\mathfrak{M}} \in E_w(A),\\
&\mathfrak{M}, w \models \langle A \rangle \varphi \quad \text{iff} \quad W \setminus \|\varphi\|^{\mathfrak{M}} \notin E_w(A),
\end{aligned}$$

where $\|\varphi\|^{\mathfrak{M}} = \{w \in W \mid \mathfrak{M}, w \models \varphi\}$ is the set of states that satisfy φ in $\mathfrak{M}$.

Example 2. *For the system from Example 1, we have for instance $w_0 \models [c](\mathsf{Sent} \sqsubseteq \mathsf{Received})$: the channel can guarantee that all sent messages will be received. However, the same property does not hold for some other states (e.g., $w_1 \not\models [c](\mathsf{Sent} \sqsubseteq \mathsf{Received})$) because the channel is memoryless and does not buffer undelivered messages. Other important properties are: $([\emptyset]\mathsf{Sent}) = \mathsf{Sent}$ (messages that are guaranteed to be labeled as sent in the next step are exactly those that have been sent until now) and $([s]\mathsf{Sent}) = \top$ (s is free to send any message); both formulas hold in every state of the system.*

The following pair of formulas demonstrates the distinction between $[A]$ as a coalitional modality vs. concept constructor. $w_0 \models ([s, c]\mathsf{Received}) = \top$: every message can be transmitted successfully if the sender and the channel cooperate; however, $w_0 \not\models [s, c](\mathsf{Received} = \top)$: s and c cannot transmit all *messages at once.*

A formula φ is *satisfiable* if there exist a model $\mathfrak{M} = \langle W, E, \mathcal{I} \rangle$ and a state $w \in W$ such that $\mathfrak{M}, w \models \varphi$. A concept C is *satisfiable* if there exist $\mathfrak{M} = \langle W, E, \mathcal{I} \rangle$ and $w \in W$ such that $C^{\mathcal{I}(w)} \neq \emptyset$. Concept D *subsumes* concept C if $C^{\mathcal{I}(w)} \subseteq D^{\mathcal{I}(w)}$ for all models $\mathfrak{M} = \langle W, E, \mathcal{I} \rangle$ and all $w \in W$. Note that concept subsumption and concept satisfiability can be reduced to formula (un)satisfiability. Concept C is satisfiable iff formula $\neg(C \sqsubseteq \bot)$ is satisfiable and concept D subsumes concept C iff formula $\neg(C \sqsubseteq D)$ is unsatisfiable. The formula $C \sqsubseteq D$ is clearly equivalent to $\neg C \sqcup D = \top$, and $C = D$ to $(\neg C \sqcup D) \sqcap (\neg D \sqcup C) = \top$. In the remainder of this paper, we will assume without loss of generality that every atomic formula is of the form $E = \top$ and we will restrict our attention to satisfiability of formulas.

Example 3. *The satisfiability problem for formula $[c](\mathsf{Sent} \sqsubseteq \mathsf{Received}) \wedge ([s]\mathsf{Sent}) = \top \wedge ([s, c]\mathsf{Received}) = \top$ asks about the existence of a model in which agent c can guarantee that all sent messages will be received, agent s is free to send any message, and every message can be transmitted successfully if s and c cooperate.*

We observe that, as models of $\mathbf{CL}_{\mathcal{ALC}}$ can be seen as a class of (possibly evolving) strategic games, the satisfiability problem for $\mathbf{CL}_{\mathcal{ALC}}$ comes very close to that of *mechanism design*, where one seeks a set of rules that guarantees desirable behavior of agents and of the whole system.

3. TABLEAUX FOR $\mathbf{CL}_{\mathcal{ALC}}$

In this section, we define structures called tableaux and show their equivalences to $\mathbf{CL}_{\mathcal{ALC}}$ models. We proceed incrementally: first, we get rid of effectivity functions and then we define more useful abstractions of constant domain models. The structure we get at the end which is called a locally correct tableau is almost directly mappable to the data structure that the algorithm uses. Such abstractions of models are commonly used in devising decision procedures [7].

The way we proceed, and the proofs we make along the way, are very similar to the work of Lutz et al. [8] which also establishes a methodology for designing tableau decision procedures for modal DLs with constant domains. Our main deviation point is that Lutz et al. utilize constraint systems (i.e., the data structures directly used in the tableau algorithm) from the beginning whereas we postpone the introduction of constraint systems until later. This is the result of using the tableau abstraction.

To reduce the number of tableau properties, we assume all formulas and concepts to be in *negation normal form* (NNF), i.e., negation signs can appear only in front of atomic formulas and concept names. Every formula (and concept) can be transformed into an equivalent one in NNF by making use of de Morgan's laws, the duality between value restrictions and full existential quantifications, and between modal operators. The NNFs of a formula φ and a concept C are denoted by $\dot{\neg}\varphi$ and $\dot{\neg}C$, respectively.

For a $\mathbf{CL}_{\mathcal{ALC}}$ formula φ, denote by

- $con(\varphi)$ the set of all concepts occurring in φ,
- $rol(\varphi)$ the set of all role names occurring in φ,
- $\mathbb{A}\text{gt}$ the set of all agents occurring in φ,
- $for(\varphi)$ the set of all subformulas of φ,
- $con^{\doteq}(\varphi) = con(\varphi) \cup \{\doteq C \mid C \in con(\varphi)\}$,
- $for^{+}(\varphi) = for(\varphi) \cup \{[\emptyset]\vartheta \mid \langle \mathbb{A}\text{gt}\rangle\vartheta \in for(\varphi)\}$,
- $con^{+}(\varphi) = con^{\doteq}(\varphi) \cup \{[\emptyset]C \mid \langle \mathbb{A}\text{gt}\rangle C \in con^{\doteq}(\varphi)\}$.

3.1 A Tableau for $\mathbf{CL}_{\mathcal{ALC}}$

DEFINITION 5. *If φ is a $\mathbf{CL}_{\mathcal{ALC}}$ formula, a pre-tableau for φ is defined to be a pentuple $\langle \Sigma, \Lambda, \mathbf{S}, \mathcal{L}, \mathcal{E}\rangle$ such that*

- *Σ is a non-empty set of states,*
- *$\Lambda : \Sigma \to 2^{for^{+}(\varphi)}$,*
- *$\mathbf{S}$ is a non-empty set of individuals,*
- *$\mathcal{L}_w : \mathbf{S} \to 2^{con^{+}(\varphi)}$,*
- *$\mathcal{E}_w : rol(\varphi) \to 2^{\mathbf{S}\times\mathbf{S}}$,*
- *there is some $w_\varphi \in \Sigma$ such that $\varphi \in \Lambda(w_\varphi)$.*

A pre-tableau T for φ must satisfy some properties so that its equivalence to a model satisfying φ can be shown. These properties make use of the structure of concepts and formulas. The problem is that modal concepts always interact with modal formulas and defining the same property three times (once for a set of modal concepts, once for a set of modal formulas, and once for a set of modal concepts and modal formulas) is unnecessary. Therefore, we will use some notational convenience.

DEFINITION 6. *Let $\langle \Sigma, \Lambda, \mathbf{S}, \mathcal{L}, \mathcal{E}\rangle$ be a pre-tableau for φ. For a state $w \in \Sigma$, the set Φ_w is defined as*

$$\Phi_w = \{\vartheta \mid \vartheta \in \Lambda(w)\} \cup \{s : C \mid C \in \mathcal{L}_w(s) \text{ and } s \in \mathbf{S}\}.$$

α and β are placeholders for elements of a set of the form Φ_w. The expression $[A]\alpha$ is either equal to some $[A]\vartheta$ or $s : [A]C$, and $\langle A\rangle\alpha$ to some $\langle A\rangle\vartheta$ or $s : \langle A\rangle C$. If $[A]\alpha = [A]\vartheta$ or $\langle A\rangle\alpha = \langle A\rangle\vartheta$, then $\alpha = \vartheta$; and if $[A]\alpha = s : [A]C$ or $\langle A\rangle\alpha = s : \langle A\rangle C$, then $\alpha = s : C$.

As a reader with tableau background would notice, the meanings of the symbols α and β in our unifying notation are different than in Smullyan's α and β notation to classify formulas. We are now in a position to define the properties of a tableau for φ.

DEFINITION 7. *Let $T = \langle \Sigma, \Lambda, \mathbf{S}, \mathcal{L}, \mathcal{E}\rangle$ be a pre-tableau for φ. T is said to be a tableau for φ if for all $w \in \Sigma$, $s, t \in \mathbf{S}$, $\vartheta, \vartheta_1, \vartheta_2 \in for^{+}(\varphi)$, $C, C_1, C_2 \in con^{+}(\varphi)$, $R \in rol(\varphi)$, $A, A_1, \ldots, A_n \subseteq \mathbb{A}\text{gt}$, it holds that:*

(P1) if $C \in \mathcal{L}_w(s)$, then $\doteq C \notin \mathcal{L}_w(s)$,

(P2) if $C_1 \sqcap C_2 \in \mathcal{L}_w(s)$, then $C_1 \in \mathcal{L}_w(s)$ and $C_2 \in \mathcal{L}_w(s)$,

(P3) if $C_1 \sqcup C_2 \in \mathcal{L}_w(s)$, then $C_1 \in \mathcal{L}_w(s)$ or $C_2 \in \mathcal{L}_w(s)$,

(P4) if $\forall R.C \in \mathcal{L}_w(s)$ and $\langle s, t\rangle \in \mathcal{E}_w(R)$, then $C \in \mathcal{L}_w(t)$,

(P5) if $\exists R.C \in \mathcal{L}_w(s)$, then there is some $s' \in \mathbf{S}$ such that $\langle s, s'\rangle \in \mathcal{E}_w(R)$ and $C \in \mathcal{L}_w(s')$,

(P6) if $C = \top \in \Lambda(w)$, then $C \in \mathcal{L}_w(s)$,

(P7) if $\neg(C = \top) \in \Lambda(w)$, then there is some $s' \in \mathbf{S}$ such that $\doteq C \in \mathcal{L}_w(s')$,

(P8) if $\vartheta \in \Lambda(w)$, then $\neg\vartheta \notin \Lambda(w)$,

(P9) if $\vartheta_1 \wedge \vartheta_2 \in \Lambda(w)$, then $\vartheta_1 \in \Lambda(w)$ and $\vartheta_2 \in \Lambda(w)$,

(P10) if $\vartheta_1 \vee \vartheta_2 \in \Lambda(w)$, then $\vartheta_1 \in \Lambda(w)$ or $\vartheta_2 \in \Lambda(w)$,

(P11) if $\langle \mathbb{A}\text{gt}\rangle\alpha \in \Phi_w$, then $[\emptyset]\alpha \in \Phi_w$,

(P12) if $[A_1]\alpha_1, \ldots, [A_n]\alpha_n \in \Phi_w$ such that $a \in A_i \cap A_j$ implies $i = j$, then there is $v \in \Sigma$ such that $\alpha_1, \ldots, \alpha_n \in \Phi_v$,

(P13) if $\langle A\rangle\alpha, [A_1]\alpha_1, \ldots, [A_n]\alpha_n \in \Phi_w$ such that $a \in A_i \cap A_j$ implies $i = j$ and $\bigcup_{i=1}^{n} A_i \subseteq A$, then there is some $v \in \Sigma$ such that $\alpha, \alpha_1, \ldots, \alpha_n \in \Phi_v$,

(P14) if $\langle A\rangle\alpha \in \Phi_w$, then there is $v \in \Sigma$ such that $\alpha \in \Phi_v$.

PROPOSITION 1. *A $\mathbf{CL}_{\mathcal{ALC}}$ formula φ is satisfiable iff there exists a tableau for φ.*

PROOF. For the *if* direction, let $T = \langle \Sigma, \Lambda, \mathbf{S}, \mathcal{L}, \mathcal{E}\rangle$ be a tableau for φ. Define for $\vartheta \in for^{+}(\varphi)$,

$$\lfloor \vartheta \rfloor^{T} = \{w \in \Sigma \mid \vartheta \in \Lambda(w)\},$$

and for $C \in con^{+}(\varphi)$ and $s \in \mathbf{S}$,

$$\lfloor C \rfloor^{T}_{s} = \{w \in \Sigma \mid C \in \mathcal{L}_w(s)\}.$$

Note that for every $w \in \Sigma$, $s \in \mathbf{S}$ if $[A]\vartheta \in \Lambda(w)$ then $\lfloor \vartheta \rfloor^{T} \neq \emptyset$, and if $[A]C \in \mathcal{L}_w(s)$ then $\lfloor C \rfloor^{T}_{s} \neq \emptyset$ because of Property (P12) in Definition 7.

As a notational convenience, let $\lfloor \alpha \rfloor^{T}$ be equal to $\lfloor \vartheta \rfloor^{T}$ if $\alpha = \vartheta$, and let it be equal to $\lfloor C \rfloor^{T}_{s}$ if $\alpha = s : C$. A coalition model $\mathfrak{M} = \langle W, E, \mathcal{I}\rangle$ in which φ is satisfied can be defined as:

1. $W = \Sigma$.
2. $E_w(A)$ is equal to V such that
   (a) *(Case $A \neq \mathbb{A}\text{gt}$)* $V = W$, or $\exists [A_1]\alpha_1, \ldots, [A_n]\alpha_n \in \Phi_w$:
      i. $A \supseteq \bigcup_{i=1}^{n} A_i$,
      ii. $\forall a \in \mathbb{A}\text{gt} : a \in A_i \cap A_j \Rightarrow i = j$,
      iii. $\bigcap_{i=1}^{n} \lfloor \alpha_i \rfloor^{T} \subseteq V$;
   (b) *(Case $A = \mathbb{A}\text{gt}$)* $\overline{V} \notin E_w(\emptyset)$.
3. $\Delta^{\mathcal{I}(w)} = \mathbf{S}$.
4. $D^{\mathcal{I}(w)} = \{s \mid D \in \mathcal{L}_w(s)\}$ for all concept names D in $con(\varphi)$.
5. $R^{\mathcal{I}(w)} = \mathcal{E}_w(R)$.

Constant domain assumption is validated by the definition of $\Delta^{\mathcal{I}(w)}$ given above. The following lemmas complete this part of the proof:

LEMMA 2. *For all $w \in W$, E_w is playable.*

LEMMA 3. *For all $w \in \Sigma$, $E \in con^{+}(\varphi)$, and $s \in \mathbf{S}$, if $E \in \mathcal{L}_w(s)$ then $s \in E^{\mathcal{I}(w)}$.*

LEMMA 4. *For every $w \in \Sigma$ and $\psi \in for^{+}(\varphi)$, if $\psi \in \Lambda(w)$ then $\mathfrak{M}, w \models \psi$.*

For the converse, a tableau for φ can trivially be constructed from a model in which φ is satisfied. □

3.2 A Quasitableau for $\mathbf{CL}_{\mathcal{ALC}}$

Representing individuals explicitly in a tableau algorithm for a modal DL is generally problematic. To the best of our knowledge, there is no such algorithm for a constant domain modal extension of $\mathcal{ALC}$ that is similar to the logic considered in this paper. For these reasons, we will use an abstraction of a tableau called quasitableau.

DEFINITION 8. *If φ is a $\mathbf{CL}_{\mathcal{ALC}}$ formula, a pre-quasitableau for φ is defined to be a tuple $\langle \Sigma, \Lambda, \mathbf{S}, \mathbf{R}, \mathcal{L}, \mathcal{E} \rangle$ such that:*

- *Σ and Λ are as given in Definition 5,*
- *$\mathbf{S}$ is a map associating with each $w \in \Sigma$ a non-empty set of concept types,*
- *$\mathbf{R}$ is a non-empty set of runs and a run r in $\mathbf{R}$ is a function associating with every $w \in \Sigma$ a concept type $r(w)$ in $\mathbf{S}(w)$,*
- *$\mathcal{L}_w : \mathbf{S}(w) \to 2^{con^+(\varphi)}$,*
- *$\mathcal{E}_w : rol(\varphi) \to 2^{\mathbf{S}(w) \times \mathbf{S}(w)}$,*
- *there is some $w_\varphi \in \Sigma$ such that $\varphi \in \Lambda(w_\varphi)$.*

Concept types can be thought of as templates for individuals, and runs as template instantiation mechanisms. The set $\mathbf{R}$ corresponds to the set $\mathbf{S}$ of individuals in a tableau. A run $r \in \mathbf{R}$ keeps track of the concept types that represent an individual in states belonging to Σ. Since $r(w)$ is defined for each $w \in \Sigma$, the individual corresponding to r is represented at each state by a concept type. This satisfies the constant domain assumption.

It will again be convenient to use a unifying notation for modal expressions. However, the structural difference between a tableau and a quasitableau requires the redefinition and also addition of some notions.

DEFINITION 9. *Let $\langle \Sigma, \Lambda, \mathbf{S}, \mathbf{R}, \mathcal{L}, \mathcal{E} \rangle$ be a pre-quasitableau for φ. For a state $w \in \Sigma$, the set Φ_w is defined as*

$$\Phi_w = \{\vartheta \mid \vartheta \in \Lambda(w)\} \cup \{s : C \mid C \in \mathcal{L}_w(s) \text{ and } s \in \mathbf{S}(w)\}.$$

Let $\Psi \subseteq \Phi_w$ be a set consisting only of expressions of the form $[A]\alpha$ and/or $\langle A \rangle \alpha$. Then the set $\mathsf{qns}(\Psi)$ is equal to $v \in \Sigma$ such that Φ_v is a superset of the union of the following sets:

1. *$\{\vartheta \mid [A]\vartheta$ (or $\langle A \rangle \vartheta) \in \Psi\}$*
2. *$\{r(v) : C \mid r(w) : [A]C$ (or $r(w) : \langle A \rangle C) \in \Psi$ and $r \in \mathbf{R}\}$*

We are now in a position to define the properties of a quasitableau for φ.

DEFINITION 10. *Let $Q = \langle \Sigma, \Lambda, \mathbf{S}, \mathbf{R}, \mathcal{L}, \mathcal{E} \rangle$ be a pre-quasitableau for φ. Q is said to be a quasitableau for φ if for all $w \in \Sigma$, $s, t \in \mathbf{S}(w)$, $\vartheta, \vartheta_1, \vartheta_2 \in for^+(\varphi)$, $C, C_1, C_2 \in con^+(\varphi)$, $R \in rol(\varphi)$, and $A, A_1, \ldots, A_n \subseteq \mathbb{A}\mathrm{gt}$, it holds that:*

(P0) there exists a run r in $\mathbf{R}$ such that $r(w) = s$,

(P1) - (P11) are as given in Definition 7 with the only difference that in the context of a quasitableau we have concept types instead of individuals,

(P12) if $\Psi = \{[A_1]\alpha_1, \ldots, [A_n]\alpha_n\} \subseteq \Phi_w$ such that $a \in A_i \cap A_j$ implies $i = j$, then $\mathsf{qns}(\Psi) \neq \emptyset$,

(P13) if $\Psi = \{\langle A \rangle \alpha, [A_1]\alpha_1, \ldots, [A_n]\alpha_n\} \subseteq \Phi_w$ such that $a \in A_i \cap A_j$ implies $i = j$ and $\bigcup_{i=1}^{n} A_i \subseteq A$, then $\mathsf{qns}(\Psi) \neq \emptyset$,

(P14) if $\Psi = \{\langle A \rangle \alpha\} \subseteq \Phi_w$, then $\mathsf{qns}(\Psi) \neq \emptyset$.

(P0) says that each concept type is in the range of some run. Properties (P12)-(P14) enforce satisfiability preserving restrictions on the runs in $\mathbf{R}$.

PROPOSITION 5. *Let φ be a $\mathbf{CL}_{\mathcal{ALC}}$ formula. There exists a quasitableau for φ iff there exists a tableau for φ.*

PROOF. Let $T = \langle \Sigma_T, \Lambda_T, \mathbf{S}_T, \mathcal{L}^T, \mathcal{E}^T \rangle$ be a tableau for φ. Then $\varphi \in \Lambda_T(w_\varphi)$ for some $w_\varphi \in \Sigma_T$. Fix $w \in \Sigma_T$. Next define equivalence relations $\sim_w$ on $\mathbf{S}_T$ by putting $s \sim_w s'$ iff $\mathcal{L}_w(s) = \mathcal{L}_w(s')$. Consider the equivalence classes modulo $\sim_w$, abbreviated by $[s]_w$. Obviously, $\{[s]_w \mid s \in \mathbf{S}_T\}$ is finite. Choose for each equivalence class $[s]_w$ a concept type $t_{[s]_w}$. Define mappings γ_w which map concept types $t_{[s]_w}$ to sets of domain objects $s \in \mathbf{S}_T$ in the obvious way i.e., $\gamma_w(t_{[s]_w}) = [s]_w$. A quasitableau $Q = \langle \Sigma_Q, \Lambda_Q, \mathbf{S}_Q, \mathbf{R}, \mathcal{L}^Q, \mathcal{E}^Q \rangle$ can be defined from T with

1. $\Sigma_Q = \Sigma_T$
2. $\Lambda_Q(w) = \Lambda_T(w)$
3. $\mathbf{S}_Q(w) = \{t_{[s]_w} \mid s \in \mathbf{S}_T\}$
4. $\mathbf{R} = \{r_s \mid s \in \mathbf{S}_T \text{ and } \forall w \in \Sigma_Q,\ r_s(w) = t_{[s]_w}\}$
5. $\mathcal{L}^Q_w(t_{[s]_w}) = \{C \mid s \in \mathbf{S}_T \text{ and } C \in \mathcal{L}^T_w(s)\}$
6. $\mathcal{E}^Q_w(R) = \{\langle t, t' \rangle \mid \exists\ s \in \gamma_w(t) \text{ and } s' \in \gamma_w(t') \text{ with } \langle s, s' \rangle \in \mathcal{E}^T_w(R)\}$.

It is easy to see that Q satisfies all properties in Definition 10. That $\varphi \in \Lambda_Q(w_\varphi)$ follows from the construction of Λ_Q.

For the converse, a tableau for φ can easily be constructed by viewing $\mathbf{R}$ as the set of individuals in the tableau. □

3.3 A Locally Correct Tableau for $\mathbf{CL}_{\mathcal{ALC}}$

It turns out that a more compact representation of a quasitableau is possible by relaxing the definition of a run. We first define this structure called a locally correct tableau. Then we show how it can be turned into a quasitableau (and vice versa).

DEFINITION 11. *If φ is a $\mathbf{CL}_{\mathcal{ALC}}$ formula, a locally correct pre-tableau for φ is defined as a hextuple $\langle \Sigma, \Lambda, \mathbf{S}, \mathbf{O}, \mathcal{L}, \mathcal{E} \rangle$ such that*

- *Σ, Λ, $\mathbf{S}$, $\mathcal{L}$, and $\mathcal{E}$ are as given in Definition 8.*
- *$\mathbf{O}$ is a non-empty set of overruns (short for overloaded runs) and an overrun $o \in \mathbf{O}$ is a function associating with every $w \in \Sigma$ a non-empty set of concept types $o(w)$ which is a subset of $\mathbf{S}(w)$,*
- *there is some $w_\varphi \in \Sigma$ such that $\varphi \in \Lambda(w_\varphi)$.*

For a state $w \in \Sigma$ and an overrun $o \in \mathbf{O}$, $|o(w)|$ is called the overloading factor of o in w.

An overrun, in contrast to a run, can associate with a state more than one concept type; thus, enabling concept types to be reused. It is this generalization that makes a locally correct tableau a more compact representation of a model than a quasitableau.

DEFINITION 12. *Let $\langle \Sigma, \Lambda, \mathbf{S}, \mathbf{O}, \mathcal{L}, \mathcal{E} \rangle$ be a locally correct pre-tableau for φ. For a state $w \in \Sigma$, the set Φ_w is defined as*

$$\Phi_w = \{\vartheta \mid \vartheta \in \Lambda(w)\} \cup \{s : C \mid C \in \mathcal{L}_w(s) \text{ and } s \in \mathbf{S}(w)\}.$$

Let $\Psi \subseteq \Phi_w$ be a set consisting only of expressions of the form $[A]\alpha$ and/or $\langle A \rangle \alpha$. Then the set $\mathsf{Ins}(\Psi)$ is equal to $v \in \Sigma$ such that for each $[A]\vartheta$ (or $\langle A \rangle \vartheta$) $\in \Psi$, $\vartheta \in \Phi_v$; and for each s with $s : [A]C$ (or $s : \langle A \rangle C$) $\in \Psi$, there exists a concept type $t \in \mathbf{S}(v)$ with $\{C \mid s : [A]C \text{ (or } s : \langle A \rangle C) \in \Psi\} \subseteq \mathcal{L}_v(t)$, $s \in o(w)$, and $t \in o(v)$.

Let o be an overrun with $o(w) = \{s, t\}$ and $w \in \Sigma$, i.e., the individual corresponding to o is represented by s and t in w. This doesn't mean that s and t are the same concept types. Properties in the following definition make use of this feature.

DEFINITION 13. *Let $T = \langle \Sigma, \Lambda, \mathbf{S}, \mathbf{O}, \mathcal{L}, \mathcal{E} \rangle$ be a locally correct pre-tableau for φ. T is said to be a locally correct tableau for φ if for all $w \in \Sigma$, $s, t \in \mathbf{S}(w)$, $\vartheta, \vartheta_1, \vartheta_2 \in for^+(\varphi)$, $C, C_1, C_2 \in con^+(\varphi)$, $R \in rol(\varphi)$, and $A, A_1, \ldots, A_n \subseteq \mathbb{A}\mathrm{gt}$, it holds that:*

(P0) there exists an overrun o in $\mathbf{O}$ such that $s \in o(w)$,

(P1) - (P11) are as given in Definition 10,

(P12) if $\Psi = \{[A_1]\alpha_1, \ldots, [A_n]\alpha_n\} \subseteq \Phi_w$ such that $a \in A_i \cap A_j$ implies $i = j$, then $\mathsf{Ins}(\Psi) \neq \emptyset$,

(P13) if $\Psi = \{\langle A \rangle \alpha, [A_1]\alpha_1, \ldots, [A_n]\alpha_n\} \subseteq \Phi_w$ such that $a \in A_i \cap A_j$ implies $i = j$ and $\bigcup_{i=1}^{n} A_i \subseteq A$, then $\mathsf{Ins}(\Psi) \neq \emptyset$,

(P14) if $\Psi = \{\langle A \rangle \alpha\} \subseteq \Phi_w$, then $\mathsf{Ins}(\Psi) \neq \emptyset$.

(P0), (P12)-(P14) are analogous to their counterparts in Definition 10. It is therefore not hard to acknowledge that a quasitableau for φ is also a locally correct tableau for φ because each run in the quasitableau can be seen as an overrun with the overloading factor of one. However, the converse does not hold because there exist cases in which we can't (immediately) define $\mathbf{R}$.

EXAMPLE 4. *Consider the locally correct tableau $T = \langle \Sigma, \Lambda, \mathbf{S}, \mathbf{O}, \mathcal{L}, \mathcal{E} \rangle$ for φ with $\Sigma = \{w, v\}$ and $\mathcal{L}_w(s) = \{[1]C, [2]D, [2,3]E\}$, $\mathcal{L}_v(s) = \{C, D\}$, $\mathcal{L}_v(t) = \{C, E\}$ (it does not matter how φ actually looks like). T is obviously a locally correct tableau, but it cannot be a quasitableau for φ: there exists no run r with $r(w) = s$, because whatever choice $r(v) = s$ or $r(v) = t$ we make, (P12) in Definition 10 does not hold. However, it is possible to modify T and convert it into a quasitableau by duplicating the state v with all the necessary mappings.*

PROPOSITION 6. *Let φ be a $\mathbf{CL}_{\mathcal{ALC}}$ formula. There exists a locally correct tableau for φ iff there is a quasitableau for φ.*

The proof of the proposition given above generalizes the observation we made in our example. The following is an immediate consequence of Propositions 1, 5, and 6.

THEOREM 7. *A $\mathbf{CL}_{\mathcal{ALC}}$ formula φ is satisfiable iff there exists a locally correct tableau for φ.*

4. TABLEAU ALGORITHM FOR $\mathbf{CL}_{\mathcal{ALC}}$

From Theorem 7, an algorithm which constructs a (finite) representation of a locally correct tableau for a $\mathbf{CL}_{\mathcal{ALC}}$ formula can be used as a decision procedure for the satisfiability of $\mathbf{CL}_{\mathcal{ALC}}$ formulas. In this section, such an algorithm is described, and we prove its termination, soundness, and completeness.

4.1 Definition of the Algorithm

Let N_V be a set of countably infinite variable names, and $<$ be the well-order relation on N_V, and let φ be a $\mathbf{CL}_{\mathcal{ALC}}$ formula. A *constraint* for φ is (i) a formula in $for^+(\varphi)$, (ii) an atom of the form $x : C$ where $x \in N_V$ and C is a concept in $con^+(\varphi)$, or (iii) an atom of the form $(x, y) : R$ where $x, y \in N_V$ and R is a role in $rol(\varphi)$. A *constraint system* S for φ is a finite, non-empty set of constraints for φ. A *completion set* $\mathbf{T}$ for φ is a set of constraint systems for φ.

In order to avoid defining analogous expansion rules for different constraints with modal operators, we will employ the α, β unifying notation that we used in the properties of tableaux.

A variable x *occurs* in S if either one of $x : C$, $(x, y) : R$, or $(y, x) : R$ is in S. x is *fresh* for S if x does not occur in S and $x > y$ for all y occurring in S. If $S \in \mathbf{T}$, then the definition of *occurs* and *fresh* are also extended for $\mathbf{T}$. We assume that when a variable x occurs in S, the constraint $x : \top$ is also in S. If $(x, y) : R \in S$ for some R, then y is called a *R-successor* of x w.r.t. S, or just a successor when R is not important.

A variable x is *blocked* by another variable y w.r.t. a constraint system S if $\{C \mid x : C \in S\} \subseteq \{D \mid y : D \in S\}$ and $y < x$. S (and therefore $\mathbf{T}$ if $S \in \mathbf{T}$) is said to contain a *clash* if for some variable x and some concept C, $\{x : C, x : \dot{\neg} C\} \subseteq S$, or if for some formula ϑ, $\{\vartheta, \dot{\neg}\vartheta\} \subseteq S$.

Let S be a constraint system for a $\mathbf{CL}_{\mathcal{ALC}}$ formula φ. The equivalence relation $\sim_S$ on the set of variables occurring in S is defined by taking $x \sim_S y$ iff $\{C \mid x : C \in S\} = \{D \mid y : D \in S\}$. The equivalence class generated by x is denoted by $[x]_S$. Finally, $\sim(S) = \{min([x]_S) : C \mid x : C \in S\} \cup \{\vartheta \mid \vartheta \in S\}$.

Let S be a constraint system for a $\mathbf{CL}_{\mathcal{ALC}}$ formula φ. $S' \subseteq S$ is called a *modal saturation* in S if S' is equal to

1. $\{[A_1]\alpha_1, \ldots, [A_n]\alpha_n\}$ such that $a \in A_i \cap A_j$ implies $i = j$,
2. $\{\langle A \rangle \alpha, [A_1]\alpha_1, \ldots, [A_n]\alpha_n\}$ such that $a \in A_i \cap A_j$ implies $i = j$ and $\bigcup_{i=1}^{n} A_i \subseteq A$, or
3. $\{\langle A \rangle \alpha\}$.

Let S be a modal saturation. Then $strip(S)$ is equal to $\{\alpha \mid [A]\alpha \text{ (or } \langle A \rangle \alpha) \in S\}$.

The tableau *expansion rules* are given in Figures 2 (local expansion rules) and 3 (the global expansion rule). A rule is *applicable* to a constraint system S if S satisfies the condition of the rule. A rule is applied to S if its action is executed due to the applicability of the rule to S.

Let φ be the $\mathbf{CL}_{\mathcal{ALC}}$ concept to be tested for satisfiability. The tableau algorithm starts with the completion set $\mathbf{T}_\varphi = \{S\}$, where $S = \{\varphi, x : \top\}$. $\mathbf{T}_\varphi$ is then expanded by repeatedly applying the rules in such a way that the global expansion rule is applied only when none of the local expansion rules is applicable to a constraint system. The

The $\mathsf{R}_{\wedge}$ rule	
Condition:	$\vartheta_1 \wedge \vartheta_2 \in S$ and $\{\vartheta_1, \vartheta_2\} \not\subseteq S$.
Action:	Set $S = S \cup \{\vartheta_1, \vartheta_2\}$.
The $\mathsf{R}_{\vee}$ rule	
Condition:	$\vartheta_1 \vee \vartheta_2 \in S$ and $\{\vartheta_1, \vartheta_2\} \cap S = \emptyset$.
Action:	Set $S = S \cup \{\psi\}$ for some $\psi \in \{\vartheta_1, \vartheta_2\}$.
The $\mathsf{R}_{\sqcap}$ rule	
Condition:	$x : C_1 \sqcap C_2 \in S$ and $\{x : C_1, x : C_2\} \not\subseteq S$.
Action:	Set $S = S \cup \{x : C_1, x : C_2\}$.
The $\mathsf{R}_{\sqcup}$ rule	
Condition:	$x : C_1 \sqcup C_2 \in S$ and $\{x : C_1, x : C_2\} \cap S = \emptyset$.
Action:	Set $S = S \cup \{x : E\}$ for some $E \in \{C_1, C_2\}$.
The $\mathsf{R}_{\exists}$ rule	
Condition:	$x : \exists R.C \in S$, x is not blocked w.r.t. S, and x has no R-successor y w.r.t. S with $y : C \in S$.
Action:	Choose a fresh y for S and set $S = S \cup \{(x, y) : R, y : C\}$.
The $\mathsf{R}_{\forall}$ rule	
Condition:	$x : \forall R.C \in S$, there is a R-successor y of x w.r.t. S with $y : C \notin S$.
Action:	Set $S = S \cup \{y : C\}$.
The $\mathsf{R}_{=}$ rule	
Condition:	$C = \top \in S$ and $x : C \notin S$ for a variable x occurring in S.
Action:	Set $S = S \cup \{x : C\}$.
The $\mathsf{R}_{\neq}$ rule	
Condition:	$\neg(C = \top) \in S$ and there is no variable x such that $x : \doteq\!C \in S$.
Action:	Choose a fresh x for S and set $S = S \cup \{x : \doteq\!\neg C\}$.
The $\mathsf{R}_{\langle \mathbb{A}\mathrm{gt} \rangle}$ rule	
Condition:	$\langle \mathbb{A}\mathrm{gt} \rangle \alpha \in S$ and $[\emptyset]\alpha \notin S$.
Action:	Set $S = S \cup \{[\emptyset]\alpha\}$.

Figure 2: Local expansion rules for $\mathbf{CL}_{\mathcal{ALC}}$.

The $\mathsf{R}_{[A\rangle}$ rule	
Condition:	$S_1, \ldots, S_n$ are all the modal saturations in S and S is not marked as finished.
Action:	Choose a fresh x for S, create sets $S'_i = {\sim}(strip(S_i) \cup \{x : \top\})$ where $1 \leq i \leq n$, add them to $\mathbf{T}$, and mark S as finished.

Figure 3: The global expansion rule for $\mathbf{CL}_{\mathcal{ALC}}$.

expansion continues until the resulting completion set contains a clash or none of the rules is applicable to it. Such a completion set is called *complete*. If the expansion rules can be applied to $\mathbf{T}_{\varphi}$ in such a way that they yield a complete, clash-free completion set, then the algorithm returns "φ is satisfiable", and "φ is unsatisfiable" otherwise.

4.2 Correctness and Termination

THEOREM 8 (TERMINATION). *When started with the initial completion set $\mathbf{T}_{\varphi}$, the tableau algorithm terminates.*

PROOF. Let $\mathbf{T}$ be the completion set for φ that is constructed by the algorithm from $\mathbf{T}_{\varphi}$ and S_j an element of $\mathbf{T}$ with $1 \leq j \leq |\mathbf{T}|$. Denote by $L_j(x)$ the set of concepts $\{C \mid x : C \in S_j\}$. The *modal depth* $md(\psi)$ of ψ is the length of the longest chain of nested modal operators in ψ (both in subformulas and subconcepts). The modal depth $md(x : C)$ of a constraint $x : C$ is defined analogously. The modal depth $md(S_j)$ of a constraint system S_j is the maximal modal depth of constraints in S_j. The following properties can easily be derived from the definition of the algorithm:

1. The expansion rules never remove constraints from constraint systems or constraint systems from the completion set.
2. The number of subsets of $con^+(\varphi)$ is $2^{con^+(\varphi)}$, hence finite.
3. $|for^+(\varphi)|$ is finite.

To prove that any sequence of rule applications is finite, it will be enough to show that there can only be finitely many constraint systems in $\mathbf{T}$ and finitely many variables in S_j. Let us first show that

(I) S_j can only have finitely many variables.

Consider all possible cases for variable introducing rules:

- $\mathsf{R}_{\exists}$: As there can only be a finite number of distinct $L_j(x)$ in S_j (by Property 2 above), a path of role successors will eventually get blocked (by Property 1). Hence the generation of a role path with infinite length is not possible.
- $\mathsf{R}_{\neq}$: As there can only be a finite number of constraints of the form $\neg(C = \top)$ in S_j (by Property 3 above), the number of $\mathsf{R}_{\neq}$ applications is limited in S_j.
- $\mathsf{R}_{[A\rangle}$: By the definition of this rule, the constraint system $S \subseteq S_j$ contains not more than $2^{con^+(\varphi)}$ distinct variables at the moment of its generation.

Now we show that the number of constraint systems in $\mathbf{T}$ should also be finite. From (I) and Property 2, we know that there are finitely many constraints of the form $x : [A_1]C$ and $y : \langle A_2 \rangle D$ in S_j. Also, the number of modal formulas in S_j is finite due to Property 3. Hence, the maximal number of constraint systems generated by the global expansion rule from S_j is finite. Let S_l be such a constraint system. Clearly, $md(S_l) < md(S_j)$. Thus, it is not possible to have an infinite chain of constraint systems starting from S_j. □

THEOREM 9 (SOUNDNESS). *If, when started with the initial completion set $\mathbf{T}_{\varphi}$ for a $\mathbf{CL}_{\mathcal{ALC}}$ formula φ, the expansion rules can be applied in such a way that they yield a complete and clash-free completion set, then there exists a locally correct tableau for φ.*

PROOF. Let $\mathbf{T}$ be the complete and clash-free completion set constructed by the tableau algorithm from $\mathbf{T}_{\varphi}$. A pentuple $T = \langle \Sigma, \Lambda, \mathbf{S}, \mathcal{L}, \mathcal{E} \rangle$ can be defined from $\mathbf{T}$ with:

1. $\Sigma = \{j \mid S_j \in \mathbf{T} \text{ for } 1 \leq j \leq |\mathbf{T}|\}$,
2. $\Lambda(j) = \{\psi \mid \psi \in S_j\}$,
3. $\mathbf{S}(j) = \{x \mid x \text{ occurs in } S_j \text{ and } x \text{ is not blocked wrt } S_j\}$,
4. $\mathcal{L}_j(x) = \{C \mid x \in \mathbf{S}(j) \text{ and } x : C \in S_j\}$,
5. $\mathcal{E}_j(R)$ is equal to $\langle x, y \rangle \in \mathbf{S}(j) \times \mathbf{S}(j)$ such that
   (a) $(x, y) : R \in S_j$, or
   (b) $(x, z) : R \in S_j$ and y blocks z.

T satisfies properties (P1)-(P11) from Definition 13 because the expansion rules are not applicable to $\mathbf{T}$ in view of its completeness. To show that properties (P0), (P12)-(P14) hold, one must inductively construct an overrun o in T. □

THEOREM 10 (COMPLETENESS). *If there exists a locally correct tableau for φ, when started with the initial completion set $\mathbf{T}_\varphi$, the expansion rules can be applied in such a way that the tableau algorithm yields a complete and clash-free completion set.*

PROOF. Let $T = \langle \Sigma, \Lambda, \mathbf{S}, \mathbf{O}, \mathcal{L}, \mathcal{E} \rangle$ be a locally correct tableau for φ. We use this tableau to guide the application of the non-deterministic rules to construct a complete and clash-free completion set for φ. Suppose that $\mathbf{T}$ is a completion set for φ. Define J as $\{j \mid S_j \in \mathbf{T} \text{ for } 1 \leq j \leq |\mathbf{T}|\}$ and say that $\mathbf{T}$ is T-*compatible* if the following holds:

1. there is a map σ from J to Σ such that if $\vartheta \in S_j$ then $\vartheta \in \Lambda(\sigma(j))$, for every $\vartheta \in for^+(\varphi)$;
2. for each $j \in J$, there is a total function π_j from the set of variables in S_j to the set of concept types in $\mathbf{S}(\sigma(j))$ such that if $x : C \in S_j$ then $C \in \mathcal{L}_{\sigma(j)}(\pi_j(x))$, and if y is a R-successor of x w.r.t. S_j then $\langle \pi_j(x), \pi_j(y) \rangle \in \mathcal{E}_{\sigma(j)}(R)$.

LEMMA 11. *If a completion set $\mathbf{T}$ for φ is T-compatible and $\mathbf{T}'$ is the result of an expansion rule (R) application to $\mathbf{T}$, then $\mathbf{T}'$ is T-compatible as well.*

Now we show that the completeness of the tableau algorithm follows from the lemma above. Let S_1 be the (initial) constraint system in $\mathbf{T}_\varphi$, and x the variable in S_1. Set $\sigma(1) = w_\varphi$ and $\pi_1(x) = s$ for a $s \in \mathbf{S}(w_\varphi)$ (such w_φ and s exist since T is a locally correct tableau for φ). It is easy to see that these functions are as needed for $\mathbf{T}_\varphi$'s T-compatibility. We know by the claim above that whenever a rule is applicable to $\mathbf{T}_\varphi$, it can be applied in a way that it maintains T-compatibility. Also, from Theorem 8, any sequence of rule applications must terminate. Thus, we have eventually a completion set $\mathbf{T}$ that is T-compatible. This completion set must be clash-free.

Suppose otherwise. Let S_j be a constraint system in $\mathbf{T}$ such that $\{x : C, x : \dot{\neg}C\} \subseteq S_j$. Then we have $\{C, \dot{\neg}C\} \subseteq \mathcal{L}_{\sigma(j)}(\pi_j(x))$ which violates Property (P1) in Definition 13. A similar argument can be made for a clash of the form $\{\vartheta, \dot{\neg}\vartheta\} \subseteq S_j$. □

5. CONCLUSIONS

In this paper, we introduce the coalitional description logic $\mathbf{CL}_{\mathcal{ALC}}$ and present a tableau decision procedure for its constant domain variant. To our best knowledge, this is the first formal study of a logic that combines DL perspective with strategic modalities. Therefore, the paper can be seen as an initiative to integrate the game-theoretic dimension with description logics. Alternatively, one can see our proposal as an attempt to extend the agenda of modal logics of strategies to reasoning about individuals and their classes without losing decidability.

We believe that our work can be useful for the semantic web community. One of the most interesting problems in this area is to discover (in an automated way) a sequence of service executions that will satisfy a user's goals. Current approaches are mainly based on standard DL subsumption testing of the desired input and output classes [9]. On the other hand, agent logics provide well studied semantics of time, action, and strategy execution, that can be used in reasoning about web services.

Our results are also interesting algorithmically. An important property regarding the optimization of our decision procedure for $\mathbf{CL}_{\mathcal{ALC}}$ is that once the application of expansion rules to a constraint system has been exhausted, the algorithm can simply discard the constraint system from its memory. This is a unique feature for a constant domain modal DL, and the major difference from the decision procedure for $\mathbf{K}_{\mathcal{ALC}}$ [8], i.e., the normal modal logic extension of $\mathcal{ALC}$. Moreover, our algorithm does not need marked variables and the non-deterministic rules which make use of the variables in [8].

6. REFERENCES

[1] F. Baader et al., editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[2] F. Baader, R. Küsters, and F. Wolter. Extensions to description logics. In Baader et al. [1], pages 219–261.

[3] F. Baader and W. Nutt. Basic description logics. In Baader et al. [1], pages 43–95.

[4] R. C. Erdur and İ. Seylan. The design of a semantic web compatible content language for agent communication. *Expert Systems*, 25(3):268–294, 2008.

[5] H. H. Hansen. Tableau games for coalition logic and alternating-time temporal logic. Master's thesis, Universiteit van Amsterdam, 2004.

[6] I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen. From $\mathcal{SHIQ}$ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics*, 1(1):7–26, 2003.

[7] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In *Proceedings of LPAR '99*, pages 161–180, 1999. Springer.

[8] C. Lutz, H. Sturm, F. Wolter, and M. Zakharyaschev. A tableau decision algorithm for modalized $\mathcal{ALC}$ with constant domains. *Studia Logica*, 72(2):199–232, 2002.

[9] M. Paolucci, T. Kawamura, T. R. Payne, and K. P. Sycara. Semantic matching of web services capabilities. In I. Horrocks and J. A. Hendler, editors, *Proceedings of ISWC*, volume 2342 of *LNCS*, pages 333–347. Springer, 2002.

[10] M. Pauly. *Logic for Social Software*. PhD thesis, University of Amsterdam, October 2001.

[11] M. Pauly. A modal logic for coalitional power in games. *Journal of Logic and Computation*, 12(1):149–166, 2002.

[12] İ. Seylan and R. C. Erdur. A tableau decision procedure for $\mathcal{ALC}$ with monotonic modal operators and constant domains. In C. Areces and S. Demri, editors, *Proceedings of M4M*, 2007.

[13] İ. Seylan and W. Jamroga. Description logic for coalitions. Technical Report IfI-08-14, Clausthal University of Technology, 2008.

[14] F. Wolter and M. Zakharyaschev. Dynamic description logics. In M. Zakharyaschev, K. Segerberg, M. de Rijke, and H. Wansing, editors, *Proceedings of AIML*, pages 431–446. CSLI Publications, 1998.

[15] M. Wooldridge, T. Agotnes, P. E. Dunne, and W. van der Hoek. Logic for automated mechanism design – a progress report. In *Proceedings of AAAI-07*, 2007.