Formalizing Multi-Agent Systems Using Action Descriptions in Single Agent Perspective

Orkunt Sabuncu[1,2] Torsten Schaub[1] Christian Schulz-Hanke[1]

[1] University of Potsdam, Germany
[2] NOVA LINCS, Universidade Nova de Lisboa, Portugal

Abstract. Logic-based representations of multi-agent systems have been extensively studied. In this work, we focus on the action language $\mathcal{BC}$ to formalize global views of MAS domains. Methodologically, we start representing the behaviour of each agent by an action description from a single agent perspective. Then, it goes through two stages that guide the modeler in composing the global view by first designating multi-agent aspects of the domain via potential conflicts and later resolving these conflicts according to the expected behaviour of the overall system. Considering that representing single agent descriptions is relatively simpler than representing multi-agent description directly, the formalization developed here is valuable from a knowledge representation perspective.

1 Introduction

Logic-based representations of multi-agent systems (MAS) have been extensively studied. Well-defined semantics of such representations allow for carrying out various reasoning tasks about MAS. Action languages [1] are among these representations of MAS. They have been successfully applied to represent and reason not only on single agent [2, 3] but also multi-agent domains [4–6].

In this work, we focus on the action language $\mathcal{BC}$ [7] to formalize MAS domains. More specifically, the formalization deals with the global view of MAS domains, which captures all knowledge about environment and capabilities of agents. It is basically modelled as a transition diagram whose vertices and edges denote states of the domain and sets of agents' actions occurring concurrently, respectively.

Unlike earlier works [5, 6, 8], where the modeler is expected to encode the global view using an action language from scratch, our formalization takes action descriptions of each agent from a single agent perspective as input and goes through two stages focusing on different aspects of the domain. Note that representing an action description of an agent with a local perspective is relatively easier than representing the whole multi-agent domain directly due to interacting concurrent actions. The first stage in our methodology establishes an intermediate action description that covers all valid states of the global view and identifies cases of the domain where actions of agent interact and have effects that are invisible from a single agent perspective. To this end, we use *potential conflict* to specify such cases formally. The second stage addresses identified potential conflicts by resolving them so that corresponding concurrent actions have the desired effect in the environment of the domain. The resolution is achieved via defeating involved dynamic laws of the intermediate description. This is possible via a syntactic transformation that makes dynamic laws of an action description defeasible. Formal properties of this transformation are given and the existence of a resolution at the second stage is proved. The resulting action description represents the transition diagram corresponding to the global view of the MAS domain.

The action language $\mathcal{BC}$ can express defeasible laws. This capacity is crucial for our formalization and has been our main motivation in selecting $\mathcal{BC}$ as the underlying action language.

The formalization, which is the novel contribution of this work, guides the modeler and structures her efforts by making multi-agent aspects of the domain explicit. To the best of our knowledge, such a methodology has not been explicitly defined and employed before.

While the scope of this work focuses on the global view of a MAS domain, it is important to emphasize that real agents residing in the environment may have autonomy via their own local view exhibiting a behaviour different than the one represented by the agent description from a single agent perspective used in the methodology. In fact they can utilize their own strategies on how to behave and even try to perform illegal actions. It is up to the MAS architecture utilizing this global view to react to these illegal actions (by simply discarding them or issuing an execution failure).

In what follows, we provide background information on the action language $\mathcal{BC}$. We also illustrate an action description describing a single Sumo agent, which is the base of the running example used throughout the paper. In Section 3, we describe our methodology of formalizing MAS with two stages. Finally, we conclude the paper with discussion that also includes future lines of research in Section 4.

2 Background

We give below syntax and semantics of the Boolean fragment of slightly modified action language $\mathcal{BC}$, as introduced in [7].

We consider a signature of Boolean *action* and *fluent symbols*, denoted by $\mathcal{A}$ and $\mathcal{F}$, respectively. In what follows, we simply refer to them as actions and fluents. Fluents are further divided into *regular* and *defined* fluents. A defined fluent is useful for representing a property that is statically determined in terms of other fluents. A *fluent literal* is a fluent a or its negation $\neg a$. Similarly, an *action literal* is an action c or its negation $\neg c$. A *static law* is an expression of the form

$$a_0 \textbf{ if } a_1, \ldots, a_m \textbf{ ifcons } a_{m+1}, \ldots, a_n \tag{1}$$

where a_i is a fluent literal for $0 \leq m \leq n$; a *dynamic law* is an expression of the form

$$a_0 \textbf{ after } a_1, \ldots, a_m \textbf{ ifcons } a_{m+1}, \ldots, a_n \tag{2}$$

where a_0 is a regular fluent literal, each of $a_1, \ldots, a_m$ is a fluent or action literal, and $a_{m+1}, \ldots, a_n$ are fluent literals. An *action description* is a finite set of of static and dynamic laws. The ifcons part of a law is important for representing defaults. Various abbreviations, such as **impossible**, **nonexecutable**, **inertial**, and **default** laws, are useful for developing succinct action descriptions. They can be rewritten in terms of static and dynamic laws (refer to [7] for their definitions).

The semantics of action descriptions is given in terms of transition systems induced by a translation into logic programs under stable models semantics [9]. To be more precise, an action description $\mathcal{D}$ and a horizon l yield a program $P_l(\mathcal{D})$ whose stable models represent all paths of length l in the transition system corresponding to $\mathcal{D}$. The signature of each program $P_l(\mathcal{D})$ consists of labeled expressions of the form $i : a$, where $i \leq l$ and a is a fluent literal, or $i < l$ and a is an action literal. As defined in [7], each program $P_l(\mathcal{D})$ consists of the following rules

1. for each static law of form (1) in $\mathcal{D}$ and $i \leq l$ a rule of form
$$i : a_0 \leftarrow i : a_1, \ldots, i : a_m, \mathit{not\ not}\ i : a_{m+1}, \ldots, \mathit{not\ not}\ i : a_n \tag{3}$$
2. for each dynamic law of form (2) in $\mathcal{D}$ and $i < l$ a rule of form
$$(i{+}1) : a_0 \leftarrow i : a_1, \ldots, i : a_m, \mathit{not\ not}\ (i+1) : a_{m+1}, \ldots, \mathit{not\ not}\ (i+1) : a_n \tag{4}$$
3. for each regular fluent f in $\mathcal{D}$ a choice rule of form
$$\{0 : f, 0 : \neg f\} \tag{5}$$
4. for each action a in $\mathcal{D}$ and $i < l$ a choice rule of form
$$\{i : a\} \tag{6}$$
5. for each fluent f in $\mathcal{D}$ and $i \leq l$ an integrity constraint of form
$$\leftarrow \{i : f, i : \neg f\} \neq 1 \tag{7}$$
6. for each action a in $\mathcal{D}$ and $i < l$ a rule of form
$$i : \neg a \leftarrow \mathit{not}\ i : a \tag{8}$$

For a set X of labeled expressions and $i \geq 0$, define $X|_i = \{a \mid i : a \in X\}$ The transition system $(S(\mathcal{D}), T(\mathcal{D}))$ induced by an action description $\mathcal{D}$ is then defined as follows. [3]

$$S(\mathcal{D}) = \{X|_0 \mid X \textit{ is a stable model of } P_0(\mathcal{D})\} \tag{9}$$
$$T(\mathcal{D}) = \{\langle X|_0 \cap \mathcal{L}, X|_0 \cap \mathcal{A}, X|_1\rangle \mid X \textit{ is a stable model of } P_1(\mathcal{D})\} \tag{10}$$

where $\mathcal{L} = \mathcal{F} \cup \{\neg f \mid f \in \mathcal{F}\}$

Note that unlike the logic program used for semantics of $\mathcal{BC}$ in [7] we name unperformed actions by explicitly generating negative action literals in (8) and consequently allow negative action literals in the after part of a dynamic law. This is critical in a

[3] Note that $X|_1 = X|_1 \cap \mathcal{L}$.

multi-agent setting to represent situations where an agent does not perform a specific action. Such a case is illustrated using the running example in Section 3.

For illustration, consider a (single) Sumo agent in a ring divided into l horizontal slots. A Sumo can move left or right to adjacent slots in the ring, and may drop out at each end. We capture this through fluents $at(A, L)$, $out(A)$ and actions $\mathit{left}(A)$, $\mathit{right}(A)$, respectively, where the variable A stands for an agent identifier[4] and $L, L' \in \{1, \dots, l\}$.

The behavior of our simple Sumo agent in an l slot ring is represented by the following action description, composed of static laws (11)–(14) and dynamic laws (15)–(21), respectively.

$$\neg at(A,L) \textbf{ if } at(A,L') \qquad (L \neq L') \tag{11}$$
$$\neg out(A) \textbf{ if } at(A,L) \tag{12}$$
$$\neg at(A,L) \textbf{ if } out(A) \tag{13}$$
$$\textbf{impossible } \neg at(A,1), \dots, \neg at(A,l), \neg out(A) \tag{14}$$
$$at(A,L) \textbf{ after } \mathit{left}(A), at(A,L') \qquad (L = L'-1) \tag{15}$$
$$at(A,L) \textbf{ after } \mathit{right}(A), at(A,L') \qquad (L = L'+1) \tag{16}$$
$$out(A) \textbf{ after } \mathit{left}(A), at(A,1) \tag{17}$$
$$out(A) \textbf{ after } \mathit{right}(A), at(A,l) \tag{18}$$
$$\textbf{nonexecutable } \mathit{left}(A) \textbf{ if } out(A) \tag{19}$$
$$\textbf{nonexecutable } \mathit{right}(A) \textbf{ if } out(A) \tag{20}$$
$$\textbf{inertial } at(A,L), out(A) \tag{21}$$

Instantiating this action description with an agent $\boldsymbol{a}$ moving in a two-slot ring where $l = 2$ yields a transition system with three states; $s_1 = \{at(\mathbf{a},1), \neg at(\mathbf{a},2), \neg out(\mathbf{a})\}$, $s_2 = \{\neg at(\mathbf{a},1), at(\mathbf{a},2), \neg out(\mathbf{a})\}$, and $s_3 = \{\neg at(\mathbf{a},1), \neg at(\mathbf{a},2), out(\mathbf{a})\}$, along with four transitions $\langle s_1, \{\mathit{right}(\mathbf{a})\}, s_2\rangle$, $\langle s_2, \{\mathit{left}(\mathbf{a})\}, s_1\rangle$, $\langle s_1, \{\mathit{left}(\mathbf{a})\}, s_3\rangle$, and $\langle s_2, \{\mathit{right}(\mathbf{a})\}, s_3\rangle$.

3 Foundations

We consider a set $\boldsymbol{A}$ of agents whose actions are governed by an environment. We formalize such multi-agent systems by means of action descriptions in $\mathcal{BC}$, one for each agent in $\boldsymbol{A}$ and descriptions capturing their interplay.

Our formalization has two stages, which will be described in the following two subsections. The first one focuses on designating aspects of the domain that do not show up from a single-agent perspective but arise once there are multiple agents. In the second stage, all these aspects are handled so that we get a global view of the overall multi-agent system.

To be more precise, the signature of a multi-agent system $(\boldsymbol{A}, \boldsymbol{c}, \boldsymbol{r})$ consists of actions $\mathcal{A}$ and fluents $\mathcal{F}$. Each agent $\boldsymbol{a} \in \boldsymbol{A}$ is represented by an action description $\mathcal{D}_{\boldsymbol{a}}$

[4] Strictly speaking, agent identifiers are obsolete in a single-agent environment but their introduction paves the way for the multi-agent setting in the next section.

over $\mathcal{A}_{\boldsymbol{a}} \subseteq \mathcal{A}$ and $\mathcal{F}_{\boldsymbol{a}} \subseteq \mathcal{F}$ such that $\mathcal{A}_{\boldsymbol{a}} \cap \mathcal{A}_{\boldsymbol{a'}} = \emptyset$ for all $\boldsymbol{a} \neq \boldsymbol{a'}$. That is, while agents may share fluents, their actions are distinct. Each action description represents the correct behavior of an agent within the environment from a single-agent perspective — specific agents may or may not behave accordingly. Components $\boldsymbol{c}$ and $\boldsymbol{r}$ are represented by action descriptions $\mathcal{D}_{\boldsymbol{c}}$ and $\mathcal{D}_{\boldsymbol{r}}$ over $\mathcal{A}$ and $\mathcal{F}$, respectively. They correspond to the first and second stage in our methodology respectively. While the role of the component $\boldsymbol{c}$ is identifying potential conflicts, the role of $\boldsymbol{r}$ is resolving such conflicts.

3.1 Identifying potential conflicts (the first stage)

Let us extend our previous example with a second Sumo $\boldsymbol{b}$. This results in two action descriptions $\mathcal{D}_{\boldsymbol{a}}$ and $\mathcal{D}_{\boldsymbol{b}}$, which are composed of laws (11)–(21) only differing in the used agent identifier, viz. $\boldsymbol{a}$ and $\boldsymbol{b}$, respectively. Both Sumos can thus move left or right to adjacent slots in the ring and drop out at each end. Also, there can only be one Sumo in a slot at a time. A Sumo who moved or is pushed out of the ring cannot return. It can, however, resist a moving opponent by moving in the reverse direction. This results in both Sumos staying in their previous slots. Similarly, if both Sumos want to move to the same slot at the same time, they bounce back and stay in their previous slots.

The mere union of all action descriptions capturing single agents in $\boldsymbol{A}$ may lack some valid states of the multi-agent system. One role of component $\boldsymbol{c}$ is to rectify this via action description $\mathcal{D}_{\boldsymbol{c}}$. In particular, it may be necessary for the action description $\mathcal{D}_{\boldsymbol{c}}$ to defeat some of the static laws of single agent descriptions. To this end, we turn static laws of single agent descriptions into default rules by introducing abnormality fluents.

Definition 1. *Let $\mathcal{D}$ be an action description in $\mathcal{BC}$. Then, we define $\tau(\mathcal{D})$ as the result of*

1. *replacing each static law of form* (1) *in $\mathcal{D}$ by*

$$a_0 \textbf{\textit{ if }} a_1, \dots, a_m \textbf{\textit{ ifcons }} a_{m+1}, \dots, a_n, \neg ab(a_0) \tag{22}$$

2. *adding for each defined fluent $ab(a_0)$ introduced in* (22) *a rule of form*

$$\textbf{\textit{default}} \ \neg ab(a_0) \tag{23}$$

3. *copying each dynamic law without change.*

At the end of the first stage of our formalization of the global view of a multi-agent system, we get the intermediate action description:

$$\mathcal{U}_{(\boldsymbol{A},\boldsymbol{c})} = \bigcup_{a \in \boldsymbol{A}} \tau(\mathcal{D}_{\boldsymbol{a}}) \cup \mathcal{D}_{\boldsymbol{c}}$$

In our Sumo example, we do not need to defeat any static laws since the union of all single agent descriptions does not lack any states of the multi-agent system. In Section 3.3 we give an example domain where the component $\boldsymbol{c}$ has to defeat some static laws to generate some previously lacking valid states of $\mathcal{U}_{(\boldsymbol{A},\boldsymbol{c})}$.

Apart from lacking states, the mere union of agent action descriptions is prone to invalid states and transitions from a multi-agent perspective. For instance, the union of action descriptions $\mathcal{D}_{\boldsymbol{a}}$ and $\mathcal{D}_{\boldsymbol{b}}$ tolerates both Sumos in the same slots, as manifested by the states $\{at(\boldsymbol{a}, L), at(\boldsymbol{b}, L)\}$. Also, it permits both Sumos passing through each other, exhibited by the transition $\langle\{at(a, 2), at(b, 3)\}, \{right(\boldsymbol{a}), left(\boldsymbol{b})\}, \{at(a, 3), at(b, 2)\}\rangle$. Such invalid states and transitions must be ruled out by appropriate laws in $\mathcal{D}_{\boldsymbol{c}}$ to govern the interplay of $\mathcal{D}_{\boldsymbol{a}}$ and $\mathcal{D}_{\boldsymbol{b}}$. Consider the action description $\mathcal{D}_{\boldsymbol{c}}$ consisting of the laws in (24) and (25).

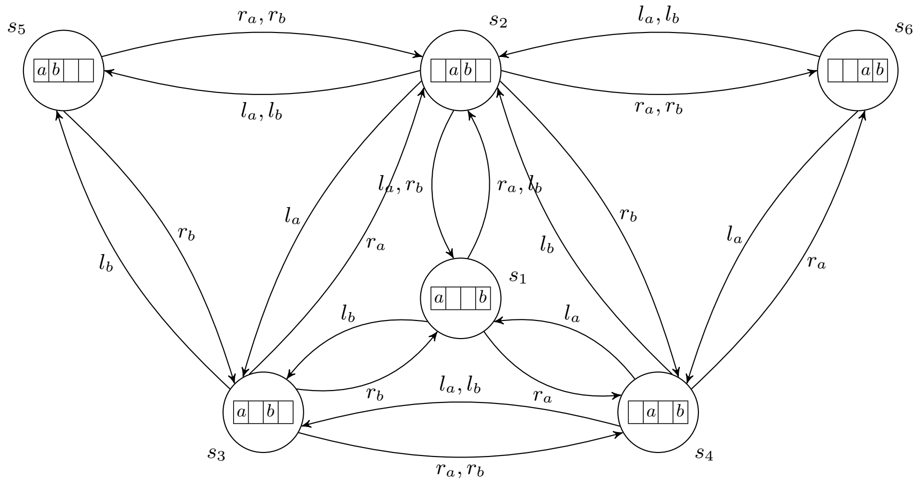
$$\neg at(A, L) \textbf{ if } at(A', L) \qquad (A \neq A') \tag{24}$$

$$\textbf{nonexecutable } right(A), left(A') \textbf{ if } at(A, L), at(A', L+1) \qquad (A \neq A') \tag{25}$$

The action description $\mathcal{U}_{(\boldsymbol{A},\boldsymbol{c})}$ induces 21 states. The part of the transition system where Sumo $\boldsymbol{a}$ is left of $\boldsymbol{b}$ is given in Figure 1.

Note that ab fluents introduced by the transformation τ are statically defined and they are false by default due to law (23). Since there are no laws in $\mathcal{D}_{\boldsymbol{c}}$ causing an ab fluent to be true in the Sumo example, there are no states of $\mathcal{U}_{(\boldsymbol{A},\boldsymbol{c})}$ in which an ab fluent holds.

Fig. 1. A part of the transition system of the union of action descriptions. Self loops with $\emptyset$ compound action are omitted for clearance. l_x and r_x abbreviate $left(x)$ and $right(x)$.



The other role of component $\boldsymbol{c}$, as stated at the beginning of this section, is to designate aspects of the domain regarding the interplay of multiple agents in the environment. To this end, we first define the notion of *potential conflict* to cover such multi-agent aspects. A potential conflict is a compound action that cannot be executed in a state in view of the laws of an action description.

Definition 2. *Let $\mathcal{D}$ be an action description in $\mathcal{BC}$, $(S(\mathcal{D}), T(\mathcal{D}))$ the corresponding transition system, and $s \in S(\mathcal{D})$.*

We define a potential conflict in s and $\mathcal{D}$ as

$$\chi_{\mathcal{D}}(s) = \{c \subseteq \mathcal{A} \mid \textit{there exists no } s' \in S(\mathcal{D}) \textit{ s.t. } \langle s, c, s' \rangle \in T(\mathcal{D})\} .$$

In our running example, there are basically 3 cases of such multi-agent aspects of the domain: (i) a Sumo can push another Sumo, (ii) Sumos can resist a push by moving in the reverse direction, and (iii) two Sumos bounce back when they want to move to the same slot at the same time. The description $\mathcal{D}_{\boldsymbol{c}}$ composed of laws (24) and (25) already identifies all potential conflicts corresponding to these 3 cases. For some representative potential conflicts of (ii) and (iii) case, consider the compound action $\{\mathit{right}(\boldsymbol{a}), \mathit{left}(\boldsymbol{b})\}$. While it can be used to switch from s_1 to s_2, it cannot be used to leave any other state in Figure 1. A transition with this compound action is ruled out in states s_5, s_2, and s_6 by (25) and in s_3 and s_4 by (24). Clearly, we have $\{\mathit{right}(\boldsymbol{a}), \mathit{left}(\boldsymbol{b})\} \in \chi_{\mathcal{U}_{(\{\boldsymbol{a},\boldsymbol{b}\},\boldsymbol{c})}}(s)$ for $s = s_2, \ldots, s_6$. Additionally, for the first case, we observe that $\{\mathit{right}(\boldsymbol{a})\} \in \chi_{\mathcal{U}_{(\{\boldsymbol{a},\boldsymbol{b}\},\boldsymbol{c})}}(s)$ and $\{\mathit{left}(\boldsymbol{b})\} \in \chi_{\mathcal{U}_{(\{\boldsymbol{a},\boldsymbol{b}\},\boldsymbol{c})}}(s)$ for $s = s_2, s_5, s_6$.

3.2 Resolving potential conflicts (the second stage)

The first stage of our formalization ends with identifying potential conflicts. Such conflicts are inadmissible in a multi-agent setting. It should be allowed for the individual agents to perform the corresponding compound action. The second stage is about preventing identified conflicts that are related to the multi-agent aspects of the domain from becoming actual conflicts. The role of the conflict resolution component $\boldsymbol{r}$ in the multi-agent system $(\boldsymbol{A}, \boldsymbol{c}, \boldsymbol{r})$ is to rectify this. To this end, the action description $\mathcal{D}_{\boldsymbol{r}}$ has to defeat some of the dynamic laws of $\mathcal{U}_{(\boldsymbol{A},\boldsymbol{c})}$.

Similar to the transformation τ, we turn dynamic laws of the description $\mathcal{U}_{(\boldsymbol{A},\boldsymbol{c})}$ into default rules by introducing an abnormality fluent using the transformation β.

Definition 3. *Let $\mathcal{D}$ be an action description in $\mathcal{BC}$.*

Then, we define $\beta(\mathcal{D})$ as the result of

1. *replacing each dynamic law of form* (2) *in $\mathcal{D}$ by*
$$a_0 \textbf{\textit{ after }} a_1, \ldots, a_m \textbf{\textit{ ifcons }} a_{m+1}, \ldots, a_n, \neg ab'(a_0) \qquad (26)$$
2. *adding for each fluent $ab'(a_0)$ introduced in* (26) *a rule of form*
$$\textbf{\textit{default }} \neg ab'(a_0) \qquad (27)$$
3. *copying each static law without change.*

Note that unlike transformation τ, abnormality fluents introduced in β are regular fluents. This allows the modeler to use such fluents in heads of dynamic laws in $\mathcal{D}_{\boldsymbol{r}}$. At the end of the second stage of the formalization we get the global view of the overall multi-agent system represented by the action description:

$$\mathcal{M}_{(\boldsymbol{A},\boldsymbol{c},\boldsymbol{r})} = \beta(\mathcal{U}_{(\boldsymbol{A},\boldsymbol{c})}) \cup \mathcal{D}_{\boldsymbol{r}}$$

Before illustrating the resolution step for our running example, we analyze the properties of transformation β and action description $\mathcal{D}_{\boldsymbol{r}}$. Due to Lemma 1, we can be sure that potential conflicts of $\mathcal{U}_{(\boldsymbol{A},\boldsymbol{c})}$ are preserved after applying β.

Lemma 1. *Let $\mathcal{D}$ be an action description in $\mathcal{BC}$. $c \in \chi_{\mathcal{D}}(s)$ iff $c \in \chi_{\beta(\mathcal{D})}(s^*)$ where $s \subseteq s^*$ and $s^* \setminus s$ has either ab' or $\neg ab'$ for each ab' fluent introduced in $\beta(\mathcal{D})$ and no other literal.*

Although the modeler knows that potential conflicts identified in the first stage of the formalization are preserved by β, she is not guided on the structure of laws needed in $\mathcal{D}_{\boldsymbol{r}}$ in order to resolve a conflict. To remedy this situation, we define a condition of a dynamic law being *covered by a compound action at a state*.

Definition 4. *A dynamic law of the form* (2) *in $\mathcal{D}$ is covered by compound action c at state $s \in S(\mathcal{D})$ iff for all $1 \leq i \leq m$ (i) $a_i \in c$ when a_i is a positive action literal, (ii) $\overline{a_i} \notin c$ when a_i is a negative action literal, and (iii) $a_i \in s$ when a_i is a fluent literal.*

Furthermore, Lemma 2 shows that laws of $\mathcal{D}_{\boldsymbol{r}}$ needed for resolving a potential conflict must be dynamic laws that are covered by the related compound action and the state. When the compound action and state are clear from the context, we may only use the statement that a law is covered.

Lemma 2. *Let $\mathcal{D}$ be an action description in $\mathcal{BC}$ over actions $\mathcal{A}$. Given a compound action $c \subseteq \mathcal{A}$ and a state $s \in S(\mathcal{D})$, $\langle s, c, s' \rangle \in T(\mathcal{D})$ iff $\langle s, c, s' \rangle \in T(\mathcal{D}')$ where $\mathcal{D}'$ is equal to $\mathcal{D}$ except all its dynamic laws not covered by c at s are taken out.*

For resolving a potential conflict, the modeler has to encode covered dynamic laws in $\mathcal{D}_{\boldsymbol{r}}$. Basically, some laws in $\mathcal{D}_{\boldsymbol{r}}$ should first defeat dynamic laws of $\beta(\mathcal{U}_{(\boldsymbol{A},\boldsymbol{c})})$ causing contradictory effects w.r.t. the desired successor state . Then additional laws in $\mathcal{D}_{\boldsymbol{r}}$ generate effect fluent literals that are previously not possible. We illustrate this methodology using our running example.

Consider the potential conflict related to two Sumos trying to move to the same slot at the same time for the state s_4 in Figure 1, i.e., $\{right(\boldsymbol{a}), left(\boldsymbol{b})\} \in \chi_{\mathcal{U}_{(\{\boldsymbol{a},\boldsymbol{b}\},\boldsymbol{c})}}(s_4)$. Considering instances of dynamic laws (15), (16), and (21), the laws (28)–(31) in $\beta(\mathcal{U}_{(\boldsymbol{A},\boldsymbol{c})})$ are covered by $\{right(\boldsymbol{a}), left(\boldsymbol{b})\}$ at s_4.

$$at(a,3) \textbf{ after } right(a), at(a,2) \textbf{ ifcons } \neg ab'(at(a,3)) \tag{28}$$
$$at(b,3) \textbf{ after } left(b), at(b,4) \textbf{ ifcons } \neg ab'(at(b,3)) \tag{29}$$
$$at(a,2) \textbf{ after } at(a,2) \textbf{ ifcons } at(a,2), \neg ab'(at(a,2)) \tag{30}$$
$$at(b,4) \textbf{ after } at(a,4) \textbf{ ifcons } at(b,4), \neg ab'(at(b,4)) \tag{31}$$

Since Sumos bounce back in this case, effect fluents $at(a,3)$ and $at(b,3)$ do not hold in the desired successor state. Hence, we need to defeat the laws (28) and (29) by causing abnormality fluents $ab'(at(a,3))$ and $ab'(at(b,3))$ to be true in order to resolve the potential conflict. In the general case, laws (32) and (33) in $\mathcal{D}_{\boldsymbol{r}}$ cause these abnormality fluents and resolve all potential conflicts related to Sumos trying to move to the same slot at the same time. Considering these potential conflicts, notice that laws (32) and

(33) are covered by respective compound actions at respective states as formally stated in Lemma 2.

$$ab'(at(A, L+1)) \textbf{ after } \mathit{right}(A), \mathit{left}(A'), at(A, L), at(A', L+2),$$
$$\neg \mathit{left}(A), \neg \mathit{right}(A') \qquad (A \neq A') \tag{32}$$
$$ab'(at(A', L+1)) \textbf{ after } \mathit{right}(A), \mathit{left}(A'), at(A, L), at(A', L+2),$$
$$\neg \mathit{left}(A), \neg \mathit{right}(A') \qquad (A \neq A') \tag{33}$$

Next, we resolve the potential conflicts related to a Sumo pushing another Sumo. For example, consider $\{\mathit{right}(\boldsymbol{a})\} \in \chi_{\mathcal{U}_{(\{\boldsymbol{a},\boldsymbol{b}\},\boldsymbol{c})}}(s_2)$. In such a case, at the desired successor state the pushed Sumo moves one slot in the direction of push (and maybe pushed out of the ring if he is at a border slot). Dynamic laws (34)–(37) in $\mathcal{D}_{\boldsymbol{r}}$ resolve such potential conflicts.

$$at(A', L+2) \textbf{ after } at(A, L), at(A', L+1), \mathit{right}(A), \neg \mathit{left}(A')$$
$$(A \neq A', L+1 < 4) \tag{34}$$
$$at(A, L-1) \textbf{ after } at(A, L), at(A', L+1), \mathit{left}(A'), \neg \mathit{right}(A)$$
$$(A \neq A', L > 1) \tag{35}$$
$$out(A') \textbf{ after } at(A, L), at(A', L+1), \mathit{right}(A), \neg \mathit{left}(A')$$
$$(A \neq A', L = 3) \tag{36}$$
$$out(A) \textbf{ after } at(A, L), at(A', L+1), \mathit{left}(A'), \neg \mathit{right}(A)$$
$$(A \neq A', L = 1) \tag{37}$$

Note that unlike the previous case, we have not defeated some covered laws by explicitly causing some ab' fluents to be true. Actually, related literals in the successor state caused by laws (34)–(37) defeat contradictory covered laws (in this case laws of intertia for the pushed Sumo) via indirect effects.

The last case concerns the potential conflicts related to two Sumos trying to push each other at the same time. In the successor state Sumos should be in their previous slots. The laws (38)–(40) in $\mathcal{D}_{\boldsymbol{r}}$ defeat contradictory covered laws, which represent effects of movement, by causing related ab' fluents to be true.

$$ab'(at(A, L+1)) \textbf{ after } \mathit{right}(A), \mathit{left}(A'), at(A, L), at(A', L+1),$$
$$\neg \mathit{left}(A), \neg \mathit{right}(A') \qquad (A \neq A') \tag{38}$$
$$ab'(at(A', L)) \textbf{ after } \mathit{right}(A), \mathit{left}(A'), at(A, L), at(A', L+1),$$
$$\neg \mathit{left}(A), \neg \mathit{right}(A') \qquad (A \neq A') \tag{39}$$
$$ab'(n(A, A', L)) \textbf{ after } \mathit{right}(A), \mathit{left}(A'), at(A, L), at(A', L+1),$$
$$\neg \mathit{left}(A), \neg \mathit{right}(A') \qquad (A \neq A') \tag{40}$$

The law (40) defeats the law in $\beta(\mathcal{U}_{(\boldsymbol{A},\boldsymbol{c})})$ that is β transformed version of (25). In principle such nonexecutable laws are abbreviations of pairs of dynamic laws with contradictory head fluents [7]. We assume that these head literals are unique for each nonexecutable law in order to avoid overly defeating such laws in the second stage. For an action description, this can be achieved by introducing some fresh fluents for each of such dynamic law pairs. Otherwise, a slightly modified version of β may generate a unique id in the introduced abnormality fluent for each nonexecutable law.[5] For instance, in our Sumo example β adds the $\neg ab'(n(A, A', L))$ abnormality fluent to the ifcons part when transforming dynamic laws abbreviated by (25) in a uniquely identified fashion.

Proposition 1 guarantees that a potential conflict in $\mathcal{U}_{(\boldsymbol{A},\boldsymbol{c})}$ can always be resolved in $\mathcal{M}_{(\boldsymbol{A},\boldsymbol{c},\boldsymbol{r})}$. The resolution used in the proposition is clearly not the only way and may also not be the pragmatic way to resolve a potential conflict (for instance, a more concise way has already been illustrated in the Sumo example). However, it forms a basic guideline as a general methodology on resolving conflicts, i.e., defeating dynamic laws causing contradictory effects and generating desired effects in the successor state.

Proposition 1. *Let $\mathcal{D}$ be an action description in $\mathcal{BC}$ over actions $\mathcal{A}$ and $c \subseteq \mathcal{A}$ be a compound action. If $c \in \mathcal{X}_{\mathcal{D}}(s)$, then for any state $s' \in S(\mathcal{D})$, $\langle s^*, c, s' \cup d\rangle \in T(\beta(\mathcal{D}) \cup \mathcal{R})$ such that $s \subseteq s^*$ and $s^* \setminus s$ has either ab' or $\neg ab'$ for each ab' fluent introduced in $\beta(\mathcal{D})$ and no other literal; $\mathcal{R}$ is a set of dynamic laws covered by c at s of the form*

$$ab'(a_0)\ \textbf{\textit{after}}\ a_1, \ldots, a_m \tag{41}$$

for each dynamic law in $\mathcal{D}$ that is covered by c at s, where $ab'(a_0)$ is introduced in $\beta(\mathcal{D})$ for the covered law in $\mathcal{D}$, and of the form

$$f\ \textbf{\textit{after}}\ a_1, \ldots, a_m \tag{42}$$

for each literal $f \in s'$; and set d is composed of positive ab' literals that appear in heads of laws (41) *in $\mathcal{R}$ and negative ab' literals for all the rest ab' fluent symbols introduced in $\beta(\mathcal{D})$.*

Let $\mathcal{D}_{\boldsymbol{r}}$ be composed of laws (32)–(40). At the end of the second stage, the action description $\mathcal{M}_{(\boldsymbol{A},\boldsymbol{c},\boldsymbol{r})}$ gives the global view of the Sumo agents. Considering a specific potential conflict, for instance, $\{right(\boldsymbol{a}), left(\boldsymbol{b})\} \in \mathcal{X}_{\mathcal{U}_{(\{\boldsymbol{a},\boldsymbol{b}\},\boldsymbol{c})}}(s_4)$ holds given the state s_4 in Figure 1. After the resolution stage, however, it is not anymore a potential conflict of $\mathcal{M}_{(\boldsymbol{A},\boldsymbol{c},\boldsymbol{r})}$ and the transition diagram has a corresponding transition, i.e., $\langle s_4, \{right(\boldsymbol{a}), left(\boldsymbol{b})\}, s_7\rangle \in T(\mathcal{M}_{(\boldsymbol{A},\boldsymbol{c},\boldsymbol{r})})$ where the successor state $s_7 = \{at(a,2), at(b,4), ab'(at(a,3)), ab'(at(b,3))\}$ (Sumos bounce back and stay in their previous slots).

Since the ab' fluents introduced by the transformation β are regular fluents, there may be states of $\mathcal{M}_{(\boldsymbol{A},\boldsymbol{c},\boldsymbol{r})}$ that include superfluous ab' fluents. In our Sumo example, for instance, $\{at(a,1), at(b,4), ab'(at(a,3))\} \in S(\mathcal{M}_{(\boldsymbol{A},\boldsymbol{c},\boldsymbol{r})})$ holds. Such states, however, are not accessible from sound initial states. We plan to address this issue formally by augmenting our formalization with a query language that enables one to express initial states and reasoning tasks (see also Section 4 for the related future work).

[5] This also applies to impossible laws and the transformation τ.

3.3 Defeating static laws in the first stage

Recall that in the Sumo example, we have not used ab fluents introduced by τ to defeat some static laws, since all valid states of the multi-agent domain are generated by the mere union of agent descriptions of single agent perspective. Consider another domain where there are two agents next to opposite ends of a table [10]. The table is initially on the floor. Each agent may lift the table up using its respective end. Whenever an agent lifts up the table, it holds the table steady in its resulting state. The table is fully lifted when it is lifted from both ends by respective agents.

Agent $\boldsymbol{l}$ is next to the left end of the table. The behaviour of this agent can be captured through fluents $table(P)$ and the action $lift\_l$ where $P \in \{onfloor, leftup\}$. The fluent $table(leftup)$ represents the situation where the left end of the table is lifted up and the right end of it is on the floor.

The behaviour of $\boldsymbol{l}$ from a single agent perspective can be modeled by the action description $\mathcal{D}_{\boldsymbol{l}}$ that is composed of laws (43)–(48).

$$\neg table(leftup)\ \textbf{if}\ table(onfloor) \tag{43}$$
$$\neg table(onfloor)\ \textbf{if}\ table(leftup) \tag{44}$$
$$\textbf{impossible}\ \neg table(onfloor), \neg table(leftup) \tag{45}$$
$$table(leftup)\ \textbf{after}\ lift\_l \tag{46}$$
$$\textbf{nonexecutable}\ lift\_l\ \textbf{if}\ table(leftup) \tag{47}$$
$$\textbf{inertial}\ table(P) \tag{48}$$

$\mathcal{D}_{\boldsymbol{l}}$ has 2 states; $s_1 = \{table(onfloor), \neg table(leftup)\}$ and $s_2 = \{table(leftup), \neg table(onfloor)\}$. It has 3 transitions; $\langle s_1, \{\}, s_1\rangle$, $\langle s_2, \{\}, s_2\rangle$, and $\langle s_1, \{lift\_l\}, s_2\rangle$.

The behaviour of agent $\boldsymbol{r}$ is similar to that of $\boldsymbol{l}$ and the description $\mathcal{D}_{\boldsymbol{r}}$ is equal to $\mathcal{D}_{\boldsymbol{l}}$ except fluent $table(rightup)$ and action $lift\_r$ are used instead of $table(leftup)$ and $lift\_l$ respectively.

The mere union of descriptions $\mathcal{D}_{\boldsymbol{l}}$ and $\mathcal{D}_{\boldsymbol{r}}$ has two states; a state where the only positive fluent literal is $table(onfloor)$ and an invalid state where both $table(leftup)$ and $table(rightup)$ hold. Besides, it is easy to see that the state where the table is fully lifted is invisible to agents from a single agent perspective.

At the end of the first stage of our formalization, $\mathcal{U}_{(\boldsymbol{A},\boldsymbol{c})}$ must cover all valid states of the multi-agent system. Let $\mathcal{D}_{\boldsymbol{c}}$ be laws (49)–(56). Note that $\mathcal{D}_{\boldsymbol{c}}$ uses the new fluent $table(lifted)$ to cover a state that is invisible from single agent perspective. However, this needs defeating static laws (45) from descriptions $\mathcal{D}_{\boldsymbol{l}}$ and $\mathcal{D}_{\boldsymbol{r}}$. This is achieved by static laws (52) and (53) where $\neg ab(imp(l))$ and $\neg ab(imp(r))$ are abnormality fluents introduced by τ to make the corresponding impossible laws defeasible.

$$\neg table(P) \textbf{ if } table(lifted) \qquad (P \neq lifted) \tag{49}$$
$$\neg table(lifted) \textbf{ if } table(P) \qquad (P \neq lifted) \tag{50}$$
$$\neg table(P) \textbf{ if } table(P') \quad (P \neq P'; P, P' \in \{rightup, leftup\}) \tag{51}$$
$$ab(imp(l)) \tag{52}$$
$$ab(imp(r)) \tag{53}$$
$$\textbf{impossible } \neg table(onfloor), \neg table(leftup), \neg table(rightup), \neg table(lifted) \tag{54}$$
$$\textbf{nonexecutable } lift\_l \textbf{ if } table(P) \quad (P \notin \{leftup, onfloor\}) \tag{55}$$
$$\textbf{nonexecutable } lift\_r \textbf{ if } table(P) \quad (P \notin \{rightup, onfloor\}) \tag{56}$$

Dynamic laws (55) and (56) eliminate some invalid transitions so that all multi-agent aspects of the domain are identified by potential conflicts in $\mathcal{U}_{(\boldsymbol{A},\boldsymbol{c})}$. There are two cases of multi-agent aspects of the domain; the table is fully lifted when both agents lift from their respective ends at the same time, or when an agent lifts up the table from his end and the table is already lifted up from the opposite end.

$\mathcal{U}_{(\boldsymbol{A},\boldsymbol{c})}$ has 4 states, one for each position of the table. All the states satisfy literals $ab(imp(l))$ and $ab(imp(r))$. It has 6 transitions. Among the 6 transitions, 4 are from each state to itself with no performed actions. The remaining transitions are from $\{table(onfloor)\}$ to $\{table(leftup)\}$ and $\{table(rightup)\}$ by compound actions $\{lift\_l\}$ and $\{lift\_r\}$, respectively.[6]

One important remark is that the modeler may encode $\mathcal{U}_{(\boldsymbol{A},\boldsymbol{c})}$ in a more compact way. Our formalization is independent of how it is encoded. Given that $\mathcal{U}_{(\boldsymbol{A},\boldsymbol{c})}$ covers all valid states of the multi-agent domain and identifies potential conflicts for all desired multi-agent aspects of the domain, it can be used in the second stage, where potential conflicts are resolved and the global view is captured by $\mathcal{M}_{(\boldsymbol{A},\boldsymbol{c},\boldsymbol{r})}$.

For the second stage of our formalization, we only show resolution of potential conflict related to the case which is about both agents lifting from their respective ends at the same time. Observe that $\{lift\_l, lift\_r\} \in \chi_{\mathcal{U}_{(\boldsymbol{A},\boldsymbol{c})}}(s)$ where $s = \{table(onfloor)\}$.[6] The dynamic laws (57)–(59) in $\mathcal{D}_{\boldsymbol{r}}$ effectively resolve this potential conflict. While laws (58) and (59) defeat the laws in $\beta(\mathcal{U}_{(\boldsymbol{A},\boldsymbol{c})})$ related to individual effects of actions $lift\_l$ and $lift\_r$, (57) caused the effect fluent in the desired successor state.

$$table(lifted) \textbf{ after } lift\_l, lift\_r, table(onfloor) \tag{57}$$
$$ab'(table(leftup)) \textbf{ after } lift\_l, lift\_r, table(onfloor) \tag{58}$$
$$ab'(table(rightup)) \textbf{ after } lift\_l, lift\_r, table(onfloor) \tag{59}$$

4 Discussion and Future Work

We have developed a formalization for capturing global view of MAS domains. Methodologically, we start representing the behaviour of each agent by an action description

[6] Here we show only positive fluent literals satisfied in a state and omit the literals $ab(imp(l))$ and $ab(imp(r))$.

in $\mathcal{BC}$ from a single agent perspective. Then, a two-stage process guides the modeler in composing these single agent descriptions into a single description representing the global view of the overall MAS domain. While the modeler designates multi-agent aspects of the domain via potential conflicts in the first stage, she resolves these conflicts according to the expected behaviour of the overall system in the second stage. Considering that representing single agent descriptions is relatively simpler than representing multi-agent description directly, the formalization developed here is valuable from a knowledge representation perspective and is different from earlier works using action languages to represent MAS domains in a monolithic way.

Our choice of $\mathcal{BC}$ as the used action language is backed by its clean semantics based on ASP and ability to express defeasible laws. Our formalization, however, is not developed with a fixed action language in mind. One can use another action language instead and update the methodology with less effort given that it can express defeasible laws (e.g., $\mathcal{C}$ [11]).

The global view of a MAS domain is useful for carrying out reasoning tasks such as projection, planning, or postdiction. We plan to extend our formalization with a query language so that a modeler can represent such reasoning tasks. Using the query language, for example, an initial and goal state of the domain can be represented to facilitate planning for the overall system.

The formalization introduced here paves the way for a number of avenues for future work. An important one is to utilize online solving capacity of the ASP solver *clingo* [12] to develop a complete multi-agent architecture based on logical representations. The solver may control execution in the environment using the formalization of the global view. Moreover, this can be performed without restarting the solver from scratch every time it communicates with the real agents residing in the environment with the help of its online solving capacity.

References

1. Gelfond, M., Lifschitz, V.: Action languages. Electronic Transactions on Artificial Intelligence **3**(6) (1998) 193–210
2. Baral, C., Gelfond, M.: Reasoning agents in dynamic domains. In Minker, J., ed.: Logic-Based Artificial Intelligence. Kluwer Academic Publishers, Dordrecht (2000) 257–279
3. Balduccini, M., Gelfond, M.: The autonomous agent architecture: An overview. In: Working Notes of the AAAI Spring Symposium on Architectures for Intelligent Theory-Based Agents (AITA08), Proceedings of the National Conference on Artificial Intelligence (AAAI) (2008) 1–6
4. Baral, C., Gelfond, M.: Reasoning about effects of concurrent actions. The Journal of Logic Programming **31**(1) (1997) 85–117
5. Baral, C., Son, T., Pontelli, E.: Reasoning about multi-agent domains using action language C: A preliminary study. In Dix, J., Fisher, M., Novák, P., eds.: Computational Logic in Multi-Agent Systems - 10th International Workshop, CLIMA X, Hamburg, Germany, September 9-10, 2009, Revised Selected and Invited Papers. Volume 6214 of Lecture Notes in Computer Science., Springer-Verlag (2009) 46–63
6. Gelfond, G., Watson, R.: Modeling cooperative multi-agent systems. In Costantini, S., Watson, R., eds.: Proceedings of the 4th Workshop Answer Set Programming: Advances in Theory and Implementation (ASP'07). (2007) 67–81

7. Lee, J., Lifschitz, V., Yang, F.: Action language BC: Preliminary report. In Rossi, F., ed.: Proceedings of the Twenty-third International Joint Conference on Artificial Intelligence (IJCAI'13), IJCAI/AAAI Press (2013) 983–989
8. Chintabathina, S., Gelfond, M., Watson, R.: Defeasible laws, parallel actions, and reasoning about resources. In Amir, E., Lifschitz, V., Miller, R., eds.: Proceedings of the Eighth International Symposium on Logical Formalization of Commonsense Reasoning (COMMONSENSE'07), Proceedings of the National Conference on Artificial Intelligence (AAAI) (2007)
9. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In Kowalski, R., Bowen, K., eds.: Proceedings of the Fifth International Conference and Symposium of Logic Programming (ICLP'88), MIT Press (1988) 1070–1080
10. Pednault, E.: Formulating multi-agent dynamic-world problems in the classical planning framework. In Georgeff, M., Lansky, A., eds.: Reasoning about actions and plans, Morgan Kaufmann Publishers (1987) 47–82
11. Giunchiglia, E., Lifschitz, V.: An action language based on causal explanation: Preliminary report. In: Proceedings of the National Conference on Artificial Intelligence (AAAI). (1998) 623–630
12. Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T.: *Clingo* = ASP + control: Preliminary report. In Leuschel, M., Schrijvers, T., eds.: Technical Communications of the Thirtieth International Conference on Logic Programming (ICLP'14). Volume arXiv:1405.3694v1 of Theory and Practice of Logic Programming, Online Supplement. (2014) Available at http://arxiv.org/abs/1405.3694v1.