Multi-agent systems specification and certification
a situation and state calculus approach

Paula Gouveia (mpg@math.ist.utl.pt) and Jaime Ramos
(jabr@math.ist.utl.pt)
CLC - Dep. de Matemática, IST, Av. Rovisco Pais, 1049–001 Lisboa, Portugal

Abstract.
We address the topic of specifying multi-agent systems using the situation and state calculus (SSC). SSC has been proposed as an extension of the situation calculus to overcome some limitations of the usual notion of state. The envisaged multi-agent system specification framework allows the uniform treatment of both local and global properties, providing also techniques for reasoning about such specifications. When a certain intended property is not inferred from a specification, we cannot always just add to it the corresponding formula. Indeed, it is often the case that specification axioms are required to be formulae of a certain kind. The task of identifying the new axioms that should be added to the specification in order to ensure the intended property has an abductive nature. Herein, we develop abductive reasoning techniques to tackle this problem.

Keywords: Multi-agents, specification, certification, abduction, knowledge, situation calculus

AMS Subject classification : 68Q60, 68T27, 68T30

1. Introduction

Over the years there has been an increasing interest in the area of agent-based systems [2, 28]. A multi-agent system consists of a collection of autonomous interacting agents (a community of agents).

In general, when specifying a system, we are interested not only in a description of the system, but also in ensuring that the system fulfills certain requirements. Hence a formal language endowed with a suitable deductive system should be used for specification. The use of logic for specifying and reasoning about such systems has received much attention [13, 9, 2, 4, 3]. Herein, we address the issue of the specification and certification of multi-agent systems using a logic based approach, the situation (and state) calculus.

There are many advantages in considering modular system specifications, i.e., specifications consisting only of a collection of local specifications (one for each agent) together with a specification of the interaction among agents. The information kept by the agent is encapsulated, i.e., only the agent can access it and change it by performing one of its actions. These modular specifications allow modular verification techniques. First, we can deduce properties of a single agent using only its specification (local level). These local properties can then be imported to the community (global) level. Clearly, it is easier to prove a property of an agent reasoning only at local level and this is an advantage of the use of this modular approach. However, some properties can only be expressed at global level and, as a consequence, can only be proved at this level. Herein, we propose such a modular approach to multi-agent system specification providing a uniform treatment of local and global reasoning, in the sense that techniques developed for local reasoning can also be applied to global reasoning.

Often, specifications are restricted to certain kinds of formulae [23, 10, 9]. Accordingly, we consider local specifications to include formulae describing initial conditions, the effects of actions on attributes and preconditions (enabling conditions) on action occurrence. Interaction among agents corresponds to *synchronous action calling*, i.e., the occurrence of an action in one agent triggers the simultaneous occurrence of an action in another agent. Interaction specification consists of formulae involving only the relevant actions. Many of the ideas underlying our work have been proposed in [25, 26] in the context of temporal logic.

When reasoning about a specification, it may happen that some envisaged property is not inferred from it. Since the specification only includes certain kinds of formulae, the formula characterizing this property cannot just be added, in general, to the specification. Hence, we need to provide an enriched specification from which the property now follows. Traditional certification approaches do not usually provide help on how a specification can be enriched. The task of identifying new specification axioms that should be added, in order to ensure the intended property, has an abductive nature [18]. Herein, we propose abductive reasoning techniques to tackle this problem (considering Boolean attributes), capitalizing on previous work in the context of temporal logic [8]. To our knowledge, this abductive approach to certification is a novel one in this context of multi-agent system specification. We start by working with a simplified version of agent where we do not consider some of the agent features such as knowledge. Later on, we extend this notion in order to cope with the knowledge dimension following, at local level, the ideas presented in [24, 9]. Then, we extend these knowledge features to multi-agent systems. In the context of the situation calculus, other approaches to this problem have been proposed [13], but not following this modular strategy for system specification.

As referred, we adopt an extended version of the situation calculus. The situation calculus was proposed in [16] as a formalism for specifying dynamic systems. Throughout the years several refinements and extensions have been proposed in order to cope with problems like temporal reasoning, concurrency, etc. [6, 19, 9, 14]. Temporal aspects are particularly relevant to our purposes. A significant extension in this direction is proposed in [23, 10], where the situation calculus is extended with an ordering relation on situations, endowing them with a temporal dimension. However, this temporal dimension prevents situations from being reached more than once and so situations loose the underlying notion of state (a state can be reached several times). To cope with this problem, the state of a system at a given situation is usually assumed to be characterized by the set of fluents that hold in that situation. For instance, in the fluent calculus [27], the situation calculus is extended with a function *State* that defines the state of a situation as a set of fluents. However, this notion of state is not adequate to our purposes. Consider a *stack* with actions *push* and *pop* and fluent *top* (with the usual meaning). Let s_1 and s_2 be situations with different states (*top* holds with different values). Next, consider situations $s_3{=}do(push(3), s_1)$ and $s_4{=}do(push(3), s_2)$, that clearly have the same state (the *top* is the same). Finally, consider situations $s_5 = do(pop, s_3)$ and $s_6 = do(pop, s_4)$. Again, we have two situations with different states (*top* holds for the same values as in s_1 and s_2), which raises a problem. We reach two situations with different states (s_5 and s_6) by doing the same action (*pop*) on two situations with the same state. Since we do not want to consider nondeterministic behaviour in our systems, this notion of state is not suitable. Of course we could add more fluents describing the contents of the stack in such a way that s_3 and s_4 have different states, but then the resulting specification would not be a specification of a stack (a data structure in which we can only access its top element). In [20, 21] an extension of the situation calculus (the situation and state calculus) is proposed, where states become a part of the language just as situations, actions and fluents. A state can be seen as an equivalence class of situations sharing the same properties.

The rest of the paper is organized as follows: in Section 2 we introduce the situation and state calculus; in Section 3 we present our approach to the specification of multi-agent systems and we propose techniques for reasoning about such systems; in Section 4 we propose techniques for local and global abduction; in Section 5 we extend the notion of agent to cope with knowledge; in Section 6 we present the conclusions, compare our proposal with others and discuss future work.

2. Situation and state calculus

In this section, we introduce the situation and state calculus, a many-sorted first order language. We choose to be more formal at these early stages because later on, when defining communities of agents, it will be straightforward to characterize the underlying language, capitalizing on the already defined language of each agent.

We start by defining the notion of many-sorted first-order signature with equality, where we fix all the symbols that may be used in the language, together with the type of each symbol. Hence, a many-sorted first order signature is a tuple $\langle G, F, P, \tau_F, \tau_P \rangle$ where G is a set of sorts, F and P are sets of function and predicate symbols, and τ_F and τ_P are ranking functions, i.e., functions that assign a type to function symbols and predicate symbols, respectively. Among the sorts, and as we are aiming at defining the language of the situation and state calculus, we consider the sorts *act* for *actions*, *flt* for *fluents*, *sit* for *situations* and *stt* for *states*. We consider also the following operations: (i) the constant symbols S_0 with $\tau_F(S_0) = sit$, nil with $\tau_F(nil) = act$, and $\mathbf{t}$, $\mathbf{f}$ with $\tau_F(\mathbf{t}) = \tau_F(\mathbf{f}) = flt$; (ii) the function symbols do with $\tau_F(do) = act \times sit \to sit$, and $[\cdot]$ with $\tau_F([\cdot]) = sit \to stt$; (iii) the predicate symbols $holds$ with $\tau_P(holds) = flt \times stt$, $poss$ with $\tau_P(poss) = act \times stt$, $<$ with $\tau_P(<) = sit \times sit$, and $actual$ with $\tau_P(actual) = sit$. The symbol S_0 corresponds to the initial situation. The symbol *nil* corresponds to the idle action that is always enabled and does not change the state. The symbol $\mathbf{t}$ ($\mathbf{f}$) corresponds to a fluent that hold in every state (does not hold in any state). Although not strictly necessary, these constants will be useful later for specification purposes. The symbol *do* is used for building new situations from actions. The symbol $[\cdot]$ assigns a state to each situation. The symbol *holds* defines which fluents hold in which states The symbol *poss* characterizes the enabling of an action in a given state. The symbol $<$ is an ordering relation on situations (with a tree-like structure). The symbol *actual* selects an actual path along the tree of situations [19]. Given a signature Σ, $T_{\Sigma,g}$ denotes the set of terms of sort g over Σ and L_Σ the set of formulas over Σ. We use a, s, f and u, possibly with subscripts or superscripts, to denote variables of sorts *act*, *sit*, *flt* and *stt*, respectively.

We now present the situation and state calculus axiomatization. We start by the axioms for situations, proposed in [23, 10], where situations are temporally ordered in the sense that if a situations s is related ($<$) to a situation s' then, in order to reach s' we must first reach s. These axioms are called *foundational axioms* and are denoted by $Ax_{\mathcal{F}}$:

$$\forall a, s\, (S_0 \neq do(a, s)) \tag{1.1}$$

$$\forall a_1, a_2, s_1, s_2\; do(a_1, s_1) = do(a_2, s_2) \to (a_1 = a_2 \wedge s_1 = s_2) \tag{1.2}$$

$$\forall s\, \neg(s < S_0) \tag{1.3}$$

$$\forall a, s, s'\, (s < do(a, s')) \leftrightarrow (s \leq s') \tag{1.4}$$

$$\alpha^s_{S_0} \to (\forall a, s(\alpha \to \alpha^s_{do(a,s)}) \to \forall s\, \alpha) \tag{1.5}$$

Axioms (1.1) and (1.2) are unique name assumptions. Axiom (1.5) is an induction axiom schema. Together, they are used to generate the set of situations. Axioms (1.3) and (1.4) together with the previous one define $<$ as a partial order on situations with minimal element S_0 (as usual, we use $s \leq s'$ as an abbreviation of $s < s' \vee s = s'$).

The set $Ax_{\mathcal{S}}$ of state axioms contains the following formulae:

$$\forall a, s_1, s_2\ [s_1] = [s_2] \to [do(a, s_1)] = [do(a, s_2)] \tag{2.1}$$

$$\forall u\, \exists s\ u = [s] \tag{2.2}$$

Axiom (2.1) imposes determinacy on actions and (2.2) ensures that all states are generated from situations. We introduce also axioms for *nil* and fluents **t**, **f**. The set $Ax_{\mathcal{K}}$ of axioms for this symbols, contains:

$$\forall u\ poss(nil, u) \tag{3.1}$$

$$\forall s\ [do(nil, s)] = [s] \tag{3.2}$$

$$\forall u\ holds(\mathbf{t}, u) \tag{3.3}$$

$$\forall u\ \neg holds(\mathbf{f}, u) \tag{3.4}$$

Axioms (3.1) and (3.2) state that *nil* is always enabled and does not change the state, respectively; (3.3) and (3.4) impose that **t** holds in every state and **f** does not hold in any state, respectively.

The predicate *actual* was proposed in [19] to incorporate (discrete) time in the situation calculus. The set$Ax_{\mathcal{A}}$ of axioms for *actual* contains:

$$actual(S_0) \tag{4.1}$$

$$\forall a, s\ actual(do(a, s)) \to (actual(s) \wedge poss(a, [s])) \tag{4.2}$$

$$\forall a_1, a_2, s\ (actual(do(a_1, s)) \wedge actual(do(a_2, s))) \to a_1 = a_2 \tag{4.3}$$

$$\forall s\ actual(s) \to \exists a\ actual(do(a, s)) \tag{4.4}$$

Axiom (4.1) implies that S_0 is always part of any time line (it will correspond to the initial instant); (4.2) means that any situation $do(a, s)$ may be an *actual* situation if s was already an *actual* situation and a is enabled in the state of situation s; (4.3) and (4.4) together mean that each *actual* situation has exactly one *actual* successor situation (this not a problem because the action *nil* is always enabled). Later on the following abbreviation, proposed in [19], will be useful:

$$occurs(a, s) =_{\text{abv}} actual(do(a, s)).$$

We consider also the set $Ax_{\mathcal{U}}$ of *uniqueness name assumption axioms* and *closure axioms*, as proposed in [10]. Similar axioms have been proposed in [25, 26] in the context of temporal logic. We denote by Ax_{SSC} the set $Ax_{\mathcal{F}} \cup Ax_{\mathcal{S}} \cup Ax_{\mathcal{K}} \cup Ax_{\mathcal{A}} \cup Ax_{\mathcal{U}}$.

3. Agents and multi-agents

We introduce the notion of agent and the notion of multi-agent system (community of agents). The definition of an agent includes its signature, setting the vocabulary, and a specification, describing the behaviour. Finally, we present a derivation relation and some reasoning techniques for proving properties about the agent specification. In the sequel, we assume available some data-types. However we will not be concerned about their specification. Hence, for our purposes, it is enough to assume that we have a set of *data-type sorts* G_{dt} and that the underlying signature includes, besides G_{dt}, all the necessary symbols for manipulating them. We will also assume that reasoning about data-types is implicit in the derivation relation of the agent and of the community.

3.1. Agents

An agent is an entity with an internal state that is characterized by a set of attributes, storing the local information of the agent, and a set of actions, for accessing internal information and interacting with other agents. A signature introduces attributes and actions and a specification describes the behaviour.

3.1.1. *Signature*

To define an agent signature we need to provide a (finite) set of actions and a (finite) set of attributes, which should be disjoint. Actions and attributes may have parameters, which are defined by a typing function. These parameters may only range over the fixed data-types (G_{dt}).

DEFINITION 1. *An* agent signature *is a tuple* $ASig = \langle Act, Att, \tau \rangle$ *where* $Act \neq \emptyset$ *is a finite set of* action symbols *or* actions*,* Att *is a finite set of* attribute symbols *or* attributes*, with* $Act \cap Att = \emptyset$ *and* $\tau : Act \cup Att \to G^*_{dt}$ *is a typing function.*

The typing function assigns to each symbol a finite sequence of data-type sorts. For instance, the action *push*, described before, has one parameter of sort *nat*. Hence, $\tau(push) = nat$. When an action (or attribute) has no parameters, the typing function assigns ϵ (the empty sequence of sorts) to it, e.g., $\tau(pop) = \epsilon$ since *pop* has no parameters.

It is straightforward to induce a situation and state calculus signature $\Sigma^{ASig} = \langle F, P, \tau_F, \tau_P \rangle$ from an agent signature $ASig = \langle Act, Att, \tau \rangle$. It is enough to consider $Act \cup Att \subseteq F$ such that $\tau_F(a) = \tau(a) \rightarrow act$ for every $a \in Act$ and $\tau_F(f) = \tau(f) \rightarrow flt$ for every $f \in Att$ (these are the only symbols of sort *act* and *flt* included in the signature, and recall that we are assuming $G_{dt} \subseteq G$). For instance, the action *push*, with $\tau(push) = nat$, corresponds, in the induced signature, to a symbol with the same name and $\tau_F(push) = nat \rightarrow act$.

EXAMPLE 1. *Consider a distributed system of readers and writers [1], where there are two types of agents competing for access to a critical section: readers, "which are not required to exclude one another", and writers, "which are required to exclude every other agent".*

A reader is a simple agent. It has two actions: read, for sending a request to access the critical section, and stop, for leaving the critical section. It has one attribute r? for signaling that the agent is in the critical section. The signature is $ASig^R = \langle \{read, stop\}, \{r?\}, \tau^r \rangle$ *where* $\tau^r(a) = \epsilon$*, for* $a \in \{read, stop\}$*, and* $\tau^r(r?) = \epsilon$*.*

The writer is similar to the reader. It has two actions: write, for sending a request to access the critical section, and stop, for leaving the critical section. It has one attribute w? for signaling that the agent is in the critical section. The signature is $ASig^W = \langle \{write, stop\}, \{w?\}, \tau^w \rangle$ *where* $\tau^w(a) = \epsilon$*, for* $a \in \{write, stop\}$*, and* $\tau^w(w?) = \epsilon$*.*

We have to consider also a buffer for managing the access to the critical section. It has four actions: startR and startW for receiving a request for using the critical section from a reader and from a writer, respectively, stopR and stopW for receiving an indication that a reader or a writer is leaving the critical section. It has two attributes: nr for counting the readers currently using the critical section, and w?, that holds if there is a writer using the critical section. The signature is $ASig^B = \langle \{startR, startW, stopR, stopW\}, \{nr, w?\}, \tau^b \rangle$ *where* $\tau^b(a) = \epsilon$*, for* $a \in \{startR, startW, stopR, stopW\}$*,* $\tau^b(nr) = nat$ *and* $\tau^b(w?) = \epsilon$*.*

In the sequel, we assume an agent signature $ASig$ and denote by Σ (instead of Σ^{ASig}) the induced situation and state calculus signature.

Next, we introduce notation that will be useful later on. The set $ASC^u_{ASig} = \{holds(f, u) : f \in T_{\Sigma,flt}\}$ is the set of *atomic state constraints* for state $u \in T_{\Sigma,stt}$. The set SC^u_{ASig} of *state constraints* for state u is the set of all Boolean combinations of elements in ASC^u_{ASig} and state independent conditions on fluent parameters (e.g. $x > 0$). The set $SC_{ASig} = \bigcup_{u \in T_{\Sigma,stt}} SC^u_{ASig}$ is the set of *state constraints* over $ASig$. With this definition we ensure that a state constraint refers to exactly one state. If $\varphi \in SC^u_{ASig}$ then $\varphi_{u'} \in SC^{u'}_{ASig}$ is φ with all occurrences of u replaced by u'. Given an atomic state constraint $holds(f, u)$,

$\mathit{flt}(\mathit{holds}(f,u))$ denotes the term f of sort flt. For each $a \in T_{\Sigma,act}$, we define the set $EC^a_{ASig} = \{\forall u\ \mathit{poss}(a,u) \to q : q \in SC^u_{ASig}\}$. The formulas in EC^a_{ASig} express *necessary enabling conditions* on action a. We consider $EC_{ASig} = \bigcup_{a \in T_{\Sigma,act}} EC^a_{ASig}$, and $\mathit{cnd}(\forall u\ \mathit{poss}(a,u) \to q) = q$. Formulae in $SC_{ASig} \cup EC_{ASig}$ are important to specify safety aspects of the agent behaviour.

The effects of an action $a \in T_{\Sigma,act}$ on attributes are going to be expressed by a formula

$$\forall s\ (\mathit{occurs}(a,s) \wedge q_{[s]}{}^{f_1 \ldots f_n}_{p_1 \ldots p_n}) \to q_{[do(a,s)]} \qquad \textsf{(Val)}$$

where $f_1, \ldots, f_n \in T_{\Sigma,\mathit{flt}}$, for $n \geq 0$, are fluent terms (corresponding to distinct attributes), $q \in SC_{ASig}$ is a state constraint and $p_1, \ldots, p_n$ are Boolean combinations of elements in $T_{\Sigma,\mathit{flt}}$. Furthermore, $q^{f_1 \ldots f_n}_{p_1 \ldots p_n}$ stands for the formula obtained from q by uniformly replacing each f_i by[1] p_i. This formula expresses that q holds at the state $[do(a,s)]$ resulting from the occurrence of a, whenever q, with all the occurrences of each f_i replaced by p_i, held at the previous state $[s]$. It may seem that the state constraints in (Val) should be swapped. However, this is not the case. In particular, if q is $\mathit{holds}(f_i,u)$, for $1 \leq i \leq n$, the formula (Val) expresses that f_i holds after the occurrence of a whenever p_i held before. When q is $\neg \mathit{holds}(f_i,u)$ the formula expresses that f_i does not hold after the occurrence of a whenever p_i did not hold before. Given $f \in T_{\Sigma,\mathit{flt}} \setminus \{f_1, \ldots, f_n\}$ the formula (Val) with $q = \mathit{holds}(f,u)$ and with $q = \neg\mathit{holds}(f,u)$ together state that the occurrence of a does not change f. We can avoid the use of all these formulas considering the formula schema (where α is a formula variable):

$$\forall s\ (\mathit{occurs}(a,s) \wedge \alpha_{[s]}{}^{f_1 \ldots f_n}_{p_1 \ldots p_n}) \to \alpha_{[do(a,s)]}. \qquad \textsf{(ValS)}$$

These formula schemata are called *valuation schemata*. The q-instance of this schema is the formula $\forall s\ (\mathit{occurs}(a,s) \wedge q_{[s]}{}^{f_1 \ldots f_n}_{p_1 \ldots p_n}) \to q_{[do(a,s)]}$. In Example 2 we further illustrate the use of valuation schemata for specification. Note that valuation schemata are somehow related to successor state axioms [22].

We denote by S^a_Σ the set of all valuation schemata for a and define $S_{ASig} = \bigcup_{a \in T_{\Sigma,act}} S^a_{ASig}$ and $\iota(S,C) = \{\varphi \in L_\Sigma :\ \varphi$ is the q-instance of some $s \in S$ and $q \in C\}$, for $S \subseteq S_{ASig}$ and $C \subseteq SC_{ASig}$.

[1] We introduce the following abbreviation: given a state constraint $q \in SC^u_{Asig}$, let $h_1, \ldots, h_k$ be all the atomic state constraints occurring in q. Then

$$\mathit{holds}(q^{h_1\ \ldots\ h_k}_{\mathit{flt}(h_1)\ldots\mathit{flt}(h_k)}, u)$$

is an abbreviation of q. For instance, considering $q = \mathit{holds}(y,u) \wedge \neg\mathit{holds}(z,u)$, the corresponding abbreviation is $\mathit{holds}(y \wedge \neg z, u)$.

3.1.2. *Specification*

An agent specification consists of a set of formulae (axioms) describing the intended behaviour of the agent. We do not allow arbitrary formulae in the specification.

DEFINITION 2. *Given an agent signature* $ASig = \langle Act, Att, \tau\rangle$, *an* agent specification *is a triple* $ASpec = \langle ASig, AxF, AxS\rangle$ *where* $AxF \subseteq SC^{[S_0]}_{ASig} \cup EC_{ASig}$ *is finite and* $AxS = \bigcup_{a \in T_{\Sigma,act}} \{s_a\}$ *where* $s_a \in S^a_{ASig}$, *for each* $a \in T_{\Sigma,act}$, *and* $s_{nil} = \forall s\ (occurs(nil, s) \wedge \alpha_{[s]}) \rightarrow \alpha_{[do(nil,s)]}$.

In an agent specification, we only allow state constrains over the state of the initial situation (for establishing the initial conditions of the system), enabling conditions and exactly one valuation schema for each action. The valuation schema for the action *nil* complies with the axiom 3.2.

EXAMPLE 2. *Consider* $ASpec^R = \langle ASig^R, AxF^R, AxS^R\rangle$, *the agent specification for the reader, where* $ASig^R$ *was defined in Example 1:*

– $AxF^R = \{\neg holds(r?, [S_0]),$
$\quad \forall u\ poss(read, u) \rightarrow \neg holds(r?, u),$
$\quad \forall u\ poss(stop, u) \rightarrow holds(r?, u)\ \}$;

– $AxS^R = \{\forall s\ (occurs(read, s) \wedge \alpha_{[s]}{}^{r?}_{\boldsymbol{t}}) \rightarrow \alpha_{[do(read,s)]},$
$\quad \forall s\ (occurs(stop, s) \wedge \alpha_{[s]}{}^{r?}_{\boldsymbol{f}}) \rightarrow \alpha_{[do(stop,s)]}\ \}$.

Consider now $ASpec^W = \langle ASig^W, AxF^W, AxS^W\rangle$, *the agent specification for the writer, where* $ASig^W$ *was defined in Example 1:*

– $AxF^W = \{\neg holds(w?, [S_0]),$
$\quad \forall u\ poss(write, u) \rightarrow \neg holds(w?, u),$
$\quad \forall s\ poss(stop, u) \rightarrow holds(w?, u)\ \}$;

– $AxS^W = \{\forall s\ (occurs(write, s) \wedge \alpha_{[s]}{}^{w?}_{\boldsymbol{t}}) \rightarrow \alpha_{[do(write,s)]},$
$\quad \forall s\ (occurs(stop, s) \wedge \alpha_{[s]}{}^{w?}_{\boldsymbol{f}}) \rightarrow \alpha_{[do(stop,s)]}\ \}$.

Finally, consider $ASpec^B = \langle ASig^B, AxF^B, AxS^B\rangle$, *the agent specification for the buffer, where* $ASig^B$ *was defined in Example 1:*

– $AxF^B = \{\neg holds(w?, [S_0]), holds(nr(0), [S_0]),$
$\quad \forall u, x, x'\ (holds(nr(x), u) \wedge holds(nr(x'), u)) \rightarrow x = x',$
$\quad \forall u\ poss(startR, u) \rightarrow \neg holds(w?, u),$
$\quad \forall u\ poss(startW, u) \rightarrow (holds(nr(0), u) \wedge \neg holds(w?, u)),$

$$\forall u \; poss(stopR, u) \rightarrow (holds(nr(x), u) \wedge x > 0),$$
$$\forall u \; poss(stopW, u) \rightarrow holds(w?, u) \,\};$$

– $AxS^B = \{\forall s \; (occurs(startR, s) \wedge \alpha_{[s]}{}^{nr(x)}_{nr(x+1)}) \rightarrow \alpha_{[do(startR,s)]},$

$$\forall s \; (occurs(startW, s) \wedge \alpha_{[s]}{}^{w?}_{\boldsymbol{t}}) \rightarrow \alpha_{[do(startW,s)]},$$
$$\forall s \; (occurs(stopR, s) \wedge \alpha_{[s]}{}^{nr(x)}_{nr(x-1)}) \rightarrow \alpha_{[do(stopR,s)]},$$
$$\forall s \; (occurs(stopW, s) \wedge \alpha_{[s]}{}^{w?}_{\boldsymbol{f}}) \rightarrow \alpha_{[do(stopW,s)]} \,\}.$$

Consider the action startW of the buffer *and assume that we are interested in verifying if the attribute w? holds afterwards. If we consider the state constraint holds(w?, u) and instantiate the corresponding schema axiom for the action we obtain the formula:*

$$\forall s \; (occurs(startW, s) \wedge holds(\boldsymbol{t}, [s])) \rightarrow holds(w?, [do(startW, s)])$$

*and as holds(**t**, u) holds universally, this is equivalent to*

$$\forall s \; occurs(startW, s) \rightarrow holds(w?, [do(startW, s)])$$

as it would be expected. If, on the other hand, we are interested in effects of this action over nr we easily conclude that it is not affected:

$$\forall s \; (occurs(startW, s) \wedge holds(nr(x), [s])) \rightarrow holds(nr(x), [do(startW, s])).$$

3.1.3. *Verification*

We now present techniques for reasoning about an agent specification. We start by presenting the derivation relation induced by an agent specification and then we present some rules for reasoning about properties of the agent. Some properties have particular significance. Namely, safety properties are of utmost importance and, herein, we focus on such properties [12]. Similar rules may also be proposed for reasoning about other properties like *liveness* [11]. Formulae specifying safety properties are calledque safety formulae. We say that $\varphi \in L_\Sigma$ is a *safety formula* if it is

$$\forall s, a \; occurs(a, s) \rightarrow (p_{[s]} \rightarrow q_{[do(a,s)]})$$

or

$$\forall s \; actual(s) \rightarrow q_{[s]}$$

for some $p, q \in SC_{ASig}$. The first formula states that in every *actual* situation, if p holds then q holds in the next *actual* situation (which is a consequence of having only a single successor *actual* situation). The second formula states that q holds in every *actual* situation, that is, q is an invariant of the specification.

DEFINITION 3. *Let* $ASpec = \langle ASig, AxF, AxS \rangle$ *be an agent specification. The derivation relation* $ASpec \vdash$ *is defined by:*

- $ASpec \vdash \varphi$ *if* $\varphi \in AxF \cup \iota(AxS, SC_{ASig})$*;*
- $ASpec \vdash \varphi$ *if* $\varphi \in Ax_{SSC}$*;*
- $ASpec \vdash$ *is closed for first-order reasoning.*

A formula φ is deducible from (or is a consequence of) an agent specification *ASpec* if it can be obtained by first-order reasoning from the specific axioms for SSC and the axioms and axiom schemata (properly instantiated) of the agent specification.

DEFINITION 4. *We say that an agent specification ASpec is* consistent *if for no formula* φ *both* $ASpec \vdash \varphi$ *and* $ASpec \vdash \neg\varphi$ *hold.*

This is equivalent to requiring that $AxF \cup \iota(AxS, SC_{ASig}) \cup Ax_{SSC}$ is a consistent set (in the usual sense).

Given $a \in T_{\Sigma,act}$ and $p, q \in SC_{ASig}$, we say that q is *initially established* in *ASpec* if $ASpec \vdash q_{[S_0]}$ and we say that *a establishes q* from p in *ASpec* if $ASpec \vdash \forall s\ occurs(a, s) \to (p_{[s]} \to q_{[do(a,s)]})$. In particular, if $p = q$ then we say that q is *preserved by a*.

We now present rules for reasoning about safety properties in an agent specification *ASpec*. In general, a safety property expresses a condition that should not be violated, i.e., should hold in every state. For instance, a *reader* and a *writer* should not access the critical section simultaneously. The first of these rules, although simple, will be useful in the next section.

PROPOSITION 1. *Let* $p, q \in SC_{ASig}$*. If, for every* $A \in Act$*,*

$$ASpec \vdash \forall s, \vec{x}(occurs(A(\vec{x}), s) \wedge p_{[s]}) \to q_{[do(A(\vec{x}),s)]}$$

then $ASpec \vdash \forall s, a\ occurs(a, s) \to (p_{[s]} \to q_{[do(a,s)]})$.

Proof. Assume the hypothesis. Since *Act* is finite and A is arbitrary, it follows that $ASpec \vdash \bigwedge_{A \in Act} \forall s, \vec{x}\ (occurs(A(\vec{x}), s) \wedge p_{[s]}) \to q_{[do(A(\vec{x}),s)]}$ which, together with a closure axiom in $Ax_{\mathcal{U}}$, implies that $ASpec \vdash \forall a, s\ occurs(a, s) \to (p_{[s]} \to q_{[do(a,s)]})$.

In order to show that a state constraint holds in every state (in fact we are only interested in *legal* states, i.e., states that correspond to actual situations) all we have to do is to prove that the intended property is initially established in *ASpec* and it is preserved by every action in *Act*. A similar technique was proposed in [11] for reasoning about safety properties in the context of temporal logic.

PROPOSITION 2. *Let* $q \in SC_{ASig}$. *If*

I1. $ASpec \vdash q_{[S_0]}$;

I2. $ASpec \vdash \forall s, \vec{x}\ (occurs(a(\vec{x}), s) \wedge q_{[s]}) \rightarrow q_{[do(a(\vec{x}),s)]}$, *for all* $a \in Act$;

then $ASpec \vdash \forall s\ actual(s) \rightarrow q_{[s]}$.

Proof. Assume (I1) and (I2) and let $\varphi = actual(s) \rightarrow q_{[s]}$. Then, $\varphi^s_{S_0}$ is $actual(S_0) \rightarrow q_{[S_0]}$ and from (I1) we immediately conclude that

$$ASpec \vdash \varphi^s_{S_0}. \qquad (\star)$$

Assume $ASpec \vdash \varphi$ and let $a \in Act$ such that $ASpec \vdash occurs(do(a\vec{x}), s)$. Using axiom (4.2), $ASpec \vdash actual(s)$ holds and so $ASpec \vdash q_{[s]}$. By (I2) we conclude $ASpec \vdash q_{[do(a(\vec{x}),s)]}$, which implies $ASpec \vdash \varphi^s_{do(a(\vec{x}),s)}$. Thus $ASpec \vdash \varphi \rightarrow \varphi^s_{do(a(\vec{x}),s)}$. As the set Act is finite and a is arbitrary, it follows that

$$ASpec \vdash \bigwedge_{a \in Act} \forall \vec{x}\ \varphi \rightarrow \varphi^s_{do(a(\vec{x}),s)}$$

which, together with a closure axiom in $Ax_{\mathcal{U}}$, implies

$$ASpec \vdash \forall a, s\ \varphi \rightarrow \varphi^s_{do(a,s)}. \qquad (\star\star)$$

Using $(\star)$ and $(\star\star)$ together with axiom (1.5) we have $ASpec \vdash \forall s\ \varphi$.

We have shown that this rule is *sound* with respect to the derivation relation of an agent. It can be shown also [20] that the rule is also *complete*, in the sense that any safety property of an agent can be proved using the previous rule. So, the question arises: why the need for this rule? Because it gives a *strategy* for proving a safety formula. And furthermore, it is more straightforward to use than the induction axiom schema. We only need to prove that the property is preserved by all the action symbols in the agent signature (instead of dealing with first-order reasoning around universal quantifiers).

EXAMPLE 3. *Consider the buffer specification of Example 2. For this specification we want to establish the desired safety result that the* writer *excludes all the* readers. *This is expressed by the formula:*

$$ASpec \vdash \forall s\ actual(s) \rightarrow (holds(w?, [s]) \rightarrow holds(nr(0), [s])).$$

The proof uses Proposition 2 with $q = holds(w?, [s]) \rightarrow holds(nr(0), [s])$. *We must proof that* (I1) *and* (I2) *hold.* (I1) *is easily established, since* $w?$ *does not hold in* $[S_0]$ *(first axiom in* AxF^B*). To establish* (I2) *we must* *prove that q is preserved by the actions* $\{startR, startW, stopR, stopW\}$. *We outline the proof for startR. We need to establish:*

$$\forall s\ (occurs(startR, s) \wedge (holds(w?, [s]) \rightarrow holds(nr(0), [s]))) \rightarrow$$
$$(holds(w?, [do(startR, s)]) \rightarrow holds(nr(0), [do(startR, s)])).$$

Usually, it is enough to instantiate the schema axiom for the corresponding action with the formula we want to prove and then derive the desired property. However, we are free to instantiate it with other formulae in order to achieve the desired result. In this case, we instantiate the valuation schema for the action with $\neg holds(w?, u)$. *Hence,*

$$\forall s\ (occurs(startR, s) \wedge \neg holds(w?, [s])) \rightarrow \neg holds(w?, [do(startR, s)]) \qquad (\star)$$

Note that $ASpec^B \vdash \forall s\, occurs(startR, s) \rightarrow poss(startR, [s])$. *Using axiom four in* AxF^B, $ASpec^B \vdash \forall s\, occurs(startR, s) \rightarrow \neg holds(w?, [s])$. *Together with* $(\star)$, $ASpec \vdash \forall s\, occurs(startR, s) \rightarrow \neg holds(w?, [do(startR, s)])$, *implying the result. The remaining actions follow a similar strategy.*

The mutual exclusion property is more commonly expressed by the formula $\forall s\ actual(s) \rightarrow \neg(holds(w?, [s]) \wedge holds(nr(x), [s]) \wedge x > 0)$ *which is logically equivalent to one we have just proved.*

3.2. Communities

A multi-agent system (community) consists of a finite collection of agents that may interact among each other. The only form of interaction we allow herein is *synchronous action calling*, i.e., the occurrence of an action in one agent triggers the occurrence of an action in another agent. However, other forms of interaction may be specified in the situation and state calculus. In order to ensure modularity, the underlying language of the community should be carefully defined. In particular, we want to prevent one agent from accessing or changing the information of another agent except via action calling. Furthermore, we should also keep in mind that we want to have a uniform treatment of local and global levels. The constructions presented here are inspired by constructions proposed in [25, 26] in the context temporal logic.

3.2.1. *Signature*

As referred above, a community consists of a finite collection of agents. Hence, the underlying signature should include a set corresponding to the identity of these agents and, for each, an agent signature.

DEFINITION 5. *An* agent community signature *is a pair* $CSig = \langle ID, Sig \rangle$ *where ID is a finite set, of agent identities, and Sig is a map assigning to each* $i \in ID$ *an agent signature.*

Without loss of generality, we assume that $ID = \{1, \ldots, n\}$.

EXAMPLE 4. *Recall the agent signatures of Example 1. The envisaged community is composed of two* readers, *one* writer *and one* buffer. *The corresponding community signature* $CSig^{rw} = \langle \{1,2,3,4\}, Sig^{rw} \rangle$ *where* $Sig^{rw}(1) = Sig^{rw}(2) = ASig^{R}$, $Sig^{rw}(3) = ASig^{W}$, *and* $Sig^{rw}(4) = ASig^{B}$.

It is straightforward to induce a many-sorted first-order signature Σ^{CSig} from a community signature $CSig$. As an end result, we want the underlying language of the community to have the same properties as the ones we have at the local level. The first step is to put all the situation and state calculus signatures (Σ^i) induced by each agent signature ($Sig(i) = \langle Act^i, Att^i, \tau^i \rangle$) together. We denote by $\oplus\Sigma = \langle \oplus G, \oplus F, \oplus P, \oplus\tau_F, \oplus\tau_P \rangle$ the disjoint union of the agent signatures[2] sharing G_{dt} and the relevant data-type operations, i.e., $\oplus G = \biguplus_{i \in ID} G^i$, $\oplus F = \biguplus_{i \in ID} F^i$, $\oplus P = \biguplus_{i \in ID} P^i$, $\oplus\tau_F(i.f) = i.\tau^i_F(f)$, for every $f \in F^i$ and $\oplus\tau_P(i.p) = i.\tau^i_P(p)$, for every $p \in P^i$. The second and final step is to add the specific elements for defining the language of the community. This corresponds to the definition of the signature $\Sigma^{CSig} = \langle G, F, P, \tau_F, \tau_P \rangle$ where:

- $G = \{gact, gsit\} \cup \oplus G$;
- $F = \oplus F \cup \{\vec{S}_0, \vec{do}, \langle\cdot\rangle\} \cup \{\cdot^a_{\downarrow i}, \cdot^s_{\downarrow i} : i \in ID\}$ and $P = \oplus P$;
- τ_F is such that

$$\begin{aligned}
&\tau_F(\vec{S}_0) = gsit \;\text{ and }\; \tau_F(\vec{do}) = gact \times gsit \to gsit,\\
&\tau_F(\cdot^a_{\downarrow i}) = gact \to i.act \;\text{ and }\; \tau_F(\cdot^s_{\downarrow i}) = gsit \to i.sit,\\
&\tau_F(\langle\cdot\rangle) = \textstyle\prod_{i \in ID} i.act \to gact;\\
&\tau_F(f) = \oplus\tau_F(f) \text{ for } f \in \oplus F;
\end{aligned}$$

- $\tau_P = \oplus\tau_P$.

The underlying language of a community of agents is the first-order language defined by the induced first-order signature. In this language, $gact$ and $gsit$ are the sorts for global actions and global situations. $\vec{S}_0$ denotes the initial (global) situation of the community (that corresponds to the initial situation of all agents). The symbol $\langle\cdot\rangle$ builds global actions from actions of the agents. The symbols $\cdot^a_{\downarrow i}$ and $\cdot^s_{\downarrow i}$ are projections

[2] We assume the following notation: for each $i \in ID$, $i.x$ is the symbol in $\oplus\Sigma$ denoting the symbol x in Σ^i For instance, $i.f \in \oplus F$ denotes the function symbol $f \in F^i$ of agent $i \in ID$, and $i.act$ denotes the sort act of Σ^i in $\oplus\Sigma$. We assume also that this notation extends to types. Hence, $1.\tau^1_F(do)$ stands for $1.act \times 1.sit \to 1.sit$.

of global actions and situations into local actions and situations of the agent i, respectively. In the sequel, we overload notation and use $\cdot_{\downarrow i}$ to denote both $\cdot^{a}_{\downarrow i}$ and $\cdot^{s}_{\downarrow i}$. We distinguish the basic sorts *act*, *sit*, *flt* and *stt* to ensure that no ill-formed terms or formulas appear. For instance, in our working example, $3.do(2.read, 3.S_0)$, corresponding to agent 3 doing an action of agent 2, is not allowed. We will denote by $i.\theta$ the translation of a term or formula θ of agent i into the community language, e.g., the term $do(read, S_0)$ of agent 1 is denoted by $1.do(read, S_0)$ in the community (instead of $1.do(1.read, 1.S_0)$). This notation extends to sets. The *set of global state constraints* for *CSig* is the set SC_{CSig} of all Boolean combinations of the (translations of) the state constraints $SC_{Sig(i)}$, for $i \in ID$. Given a global state constraint q, we denote by $q_{[s]}$ the global state constraint obtained from q by replacing each atomic state constraint $i.p$ occurring in q by $i.p_{[s_{\downarrow i}]}$ and we denote by $ID(q)$ the set of all agent identities occurring in q. This notation extends to sets. Given $\Delta \subseteq SC_{CSig}$, $\Delta_{\downarrow i} = \{\theta : i.\theta \in \Delta\}$ for $i \in ID$. We need to characterize the properties of the symbols we have just introduced.

DEFINITION 6. *The* set of axioms about the community*, denoted by* $Ax_{\mathcal{C}}$ *contains the formulae:*

$$\vec{S}_{0\downarrow i} = i.S_0,\ \textit{for every } i \in ID \tag{5.1}$$

$$\forall a, s\ \vec{do}(a, s)_{\downarrow i} = i.do(a_{\downarrow i}, s_{\downarrow i}),\ \textit{for every } i \in ID \tag{5.2}$$

$$\forall a_1, \ldots, a_n \langle a_1, \ldots, a_n\rangle_{\downarrow i} = a_i\ \textit{for every } i \in ID \tag{5.3}$$

$$\forall s, s'\ (s = s' \leftrightarrow \bigwedge_{i\in ID} s_{\downarrow i} = s'_{\downarrow i}) \tag{5.4}$$

$$\forall a, a'\ (a = a' \leftrightarrow \bigwedge_{i\in ID} a_{\downarrow i} = a'_{\downarrow i}) \tag{5.5}$$

$$\forall a \bigwedge_{i\in ID} \bigvee_{A\in Act^i} \exists \vec{x}\ a_{\downarrow i} = A(\vec{x}) \tag{5.6}$$

$$\alpha^s_{\vec{S}_0} \to ((\forall a, s\ \alpha \to \varphi^s_{\vec{do}(a,s)}) \to \forall s\ \alpha) \tag{5.7}$$

Axioms (5.1),(5.2) relate global situations with local situations of agents; (5.3) relates the construction of global actions with projection; (5.4), (5.5) establish a consistent relation between global situations and actions and their local counterparts; (5.6) is a name generation condition; (5.7) is an induction schema for global situations.

We introduce abbreviations for allowing a uniform treatment of both levels. We start by defining the relation $\vec{<}$ on global situations:

$$s \vec{<} s' =_{\text{abv}} (\bigwedge_{i\in ID} i.(s_{\downarrow i} \leq s'_{\downarrow i})) \wedge (\bigvee_{i\in ID} i.(s_{\downarrow i} < s'_{\downarrow i})).$$

A global situation s precedes s' if at least one of the projections of s precedes the corresponding projection of s' and all the other projections of s either precede or are the same as the corresponding projection of s'. Next we consider the predicates $\vec{poss}$, $\vec{actual}(s)$ and $\vec{occurs}$:

$$\vec{poss}(a, [s]) =_{\text{abv}} \bigwedge_{i \in ID} i.poss(a_{\downarrow i}, [s_{\downarrow i}])$$

$$\vec{actual}(s) =_{\text{abv}} \bigwedge_{i \in ID} i.actual(s_{\downarrow i})$$

$$\vec{occurs}(a, s) =_{\text{abv}} \vec{actual}(\vec{do}(a, s)).$$

Interaction among agents is restricted to *synchronous communication*: the occurrence of an action a_1 of agent i triggers the simultaneous occurrence of an action a_2 (the target of the interaction) of agent j. This is denoted by $i.do(a_1, s) \triangleright j.do(a_2, s)$ and expressed by the formula:

$$\vec{occurs}(a, s) \rightarrow (a_{\downarrow i} = i.a_1 \rightarrow a_{\downarrow j} = j.a_2).$$

A first attempt might be to consider $i.\vec{occurs}(a_1, s_{\downarrow i}) \rightarrow j.\vec{occurs}(a_2, s_{\downarrow j})$. However, it does not express the desired interaction because, although $s_{\downarrow i}$ and $s_{\downarrow j}$ are actual situations, it is not necessarily the case that s is an actual global situation. And it is at the level of actual global situations that we will be interested in reasoning. We can further enrich interactions by considering disjunction or exclusive disjunction of two (or more) target actions, for instance, $i.do(a_1, s) \triangleright j.do(a_2, s) \vee k.do(a_3, s)$. Furthermore, we also consider *conditional action calling* restricting calling by a condition. This condition can only depend on information of the *caller*. We do not allow an agent i to call an agent j, depending on information of others (to ensure encapsulation). Hence, the condition must be a state constraint q over the caller. The calling of action a_2 of j by action a_1 of i conditioned by q is denoted by $i.q_{[s]} \rightarrow i.do(a_1, s) \triangleright j.do(a_2, s)$. As before, we also consider disjunction or exclusive disjunction of target actions. The set IL_{CSig} is the set of all *interaction formulae of CSig*. If $\varphi = i.q_{[s]} \rightarrow (i.do(a_1, s) \triangleright j.do(a_2, s) \vee k.do(a_3, s))$, the condition of the interaction formula is $cnd(\varphi) = i.q_{[s]}$, its trigger action is $tri(\varphi) = i.a_1$ and the set of target actions is $tar(\varphi) = \{j.a_2, k.a_3\}$. These definitions extend in a obvious way to other interaction formulae. In particular, if φ has no condition, $cnd(\varphi) = i.holds(\mathbf{t}, [s])$. If $tar(\varphi)$ is a singleton we often identify it with the element it contains.

3.2.2. *Specification*

We assume a community signature $CSig = \langle ID, Sig \rangle$. A community specification is composed of a local specification for each agent and a set of interaction formulas of *CSig*, hence, guaranteeing *encapsulation* (no agent can access the information of another agent except via interaction).

DEFINITION 7. *A* community specification *is* $CSpec = \langle CSig, AxF, AxS\rangle$ *where* $AxF = \{AxF_i\}_{i\in ID\cup\{0\}}$ *and* $AxS = \{AxS_i\}_{i\in ID}$ *are such that, for* $i \in ID$, $\langle Sig(i), AxF_i, AxS_i\rangle$ *is an agent specification and* $AxF_0 \subseteq IL_{CSig}$.

Interaction is specified in AxF_0. All the other sets contain local axioms (or shemata) corresponding to the specification of each agent.

EXAMPLE 5. *Recall the community signature* $CSig^{rw}$ *of Example 4 and the agent specifications of Example 2. The envisaged community specification is* $CSpec^{rw} = \langle CSig^{rw}, AxF^{rw}, AxS^{rw}\rangle$ *where* $AxF_1^{rw} = AxF_2^{rw} = AxF^R$, $AxF_3^{rw} = AxF^W$, $AxF_4^{rw} = AxF^B$ *and*

– $$\begin{aligned} AxF_0^{rw} = \{&1.do(read, s) \rhd 4.do(startR, s),\\ &2.do(read, s) \rhd 4.do(startR, s),\\ &4.do(startR, s) \rhd (1.do(read, s) \veebar 2.do(read, s)),\\ &1.do(stop, s) \rhd 4.do(stopR, s),\\ &2.do(stop, s) \rhd 4.do(stopR, s),\\ &4.do(stopR, s) \rhd (1.do(stop, s) \veebar 2.do(stop, s)),\\ &3.do(write, s) \rhd 4.do(startW, s),\\ &4.do(startW, s) \rhd 3.do(write, s),\\ &3.do(stop, s) \rhd 4.do(stopW, s),\\ &4.do(stopW, s) \rhd 3.do(stop, s)\ \}; \end{aligned}$$

– $AxS_1^{rw} = AxS_2^{rw} = AxF^R$, $AxS_3^{rw} = AxS^W$, $AxS_4^{rw} = AxS^B$.

The first three axioms in AxF_0^{rw} *express interaction between the* readers *and the* buffer*: whenever a* reader *wants to access the critical section it must call the corresponding action of the* buffer *and the* buffer *can only accept reading requests from one of the two readers (exclusively).*

3.2.3. *Verification*

We now define a derivation relation for a community specification. Then, we prove some results about global symbols. In particular, we prove that global actions and situations have the same properties as their local counterparts, which means that we can easily extend to the community local reasoning techniques.

DEFINITION 8. *Let* $CSpec = \langle CSig, AxF, AxS\rangle$ *be a community specification. The derivation relation* $CSpec \vdash$ *is defined by:*

– *CSpec* $\vdash \varphi$ *if* $\varphi \in Ax_{\mathcal{C}}$*;*

– *CSpec* $\vdash\ i.\varphi$ *if* $ASpec^i \vdash \varphi$*, for every* $i \in ID$*;*

– *CSpec* $\vdash \varphi$ *if* $\varphi \in AxF_0$*;*

– *CSpec* $\vdash$ *is closed for first-order reasoning.*

A formula is inferred from the community specification if it is a community axiom, or it is the image of a formula that can be inferred from one of the agent specifications, or it is an interaction formula, or it can be obtained from the previous by first-order reasoning.

DEFINITION 9. *We say that a community specification CSpec is* consistent *if for no formula* φ *both* $CSpec \vdash \varphi$ *and* $CSpec \vdash \neg\varphi$ *hold.*

The next results show that the global symbols behave as their local counterparts. In the sequel we assume a community specification $CSpec = \langle CSig, AxF, AxS \rangle$ with $CSig = \langle ID, Sig \rangle$.

PROPOSITION 3. *For CSpec it holds:*

$$CSpec \vdash \forall a, s\ (\vec{S}_0 \neq \vec{do}(a, s)) \tag{6.1}$$

$$CSpec \vdash \forall a_1, a_2, s_1, s_2\ \vec{do}(a_1, s_1) = \vec{do}(a_2, s_2) \rightarrow (a_1 = a_2 \wedge s_1 = s_2) \tag{6.2}$$

$$CSpec \vdash \forall s\ \neg(s < \vec{S}_0) \tag{6.3}$$

$$CSpec \vdash \forall a, s, s'\ (s < \vec{do}(a, s') \leftrightarrow s \vec{\leq} s'). \tag{6.4}$$

Proof. The results are a consequence of their local counterparts. (6.1) For each $i \in ID$, we have $ASpec^i \vdash \forall a', s'\ (S_0 \neq do(a', s'))$. By Definition 8, it follows that $CSpec \vdash i.\forall a', s'\ (S_0 \neq do(a', s'))$. Then $CSpec \vdash i.(S_0 \neq do(a_{\downarrow i}, s_{\downarrow i}))$ holds, where a, s are arbitrary variables (of sorts *gact* and *gsit*, respectively) and, using axioms (5.1) and (5.2) of $Ax_{\mathcal{C}}$, $CSpec \vdash (\vec{S}_{0\downarrow i} \neq \vec{do}(a, s)_{\downarrow i})$ also holds. This, together with axiom (5.4) of $Ax_{\mathcal{C}}$, implies $CSpec \vdash \forall a, s\ (\vec{S} \neq \vec{do}(a, s))$. Results (6.2), (6.3) and (6.4) have similar proofs.

The formulae (6.1)-(6.4) together with the community axiom (5.7) are the *community foundational axioms.* Note that global situations have the same properties as local situations. A similar result can be obtained for *actual*, using corresponding local properties of *actual.*

PROPOSITION 4. *For CSpec it holds:*

$$CSpec \vdash \vec{actual}(\vec{S}_0) \tag{7.1}$$

$$CSpec \vdash \forall a, s\ \vec{actual}(\vec{do}(a, s)) \rightarrow (\vec{actual}(s) \wedge \vec{poss}(a, [s])) \tag{7.2}$$

$$CSpec \vdash \forall a_1, a_2, s\ (\vec{actual}(\vec{do}(a_1, s)) \wedge \vec{actual}(\vec{do}(a_2, s))) \\ \rightarrow a_1 = a_2 \tag{7.3}$$

$$CSpec \vdash \forall s\ \vec{actual}(s) \rightarrow (\exists a\ \vec{actual}(\vec{do}(a, s))) \tag{7.4}$$

EXAMPLE 6. *For the community specification* $CSpec^{rw}$ *we can prove* $CSpec^{rw} \vdash \forall s\ \neg \vec{occurs}(\langle read, A, write, B\rangle, s)$ *where* A *and* B *are arbitrary actions from* $Sig(2)$ *and* $Sig(4)$*, respectively. We assume that* $\vec{occurs}(\langle read, A, write, B\rangle, s)$ *holds and derive a contradiction. Omiting details, if* $1.read$ *occurs then* $4.startR$ *must occur, i.e.,* $B = startR$ *(first interaction axiom). Also, if* $3.write$ *occurs then* $4.startW$ *must also occur, i.e.,* $B = startW$ *(seventh interaction axiom). By uniqueness of names,* $startR \neq startW$ *and we have the intended contradiction.*

This result is relevant when we generalize the proof of safety properties to communities. The proof of such properties involves proving preservation by all actions in the signature which corresponds to all the possible combinations of actions of the agents. For instance, in our working example it would be necessary to consider 135 global actions (including *nil*). However, using results similar to the previous one, we can rule out some of these actions. With the previous result alone, we can rule out 15 of these actions. In total, we can exclude 128 actions, leaving only 7 relevant global actions that can occur: $\langle read, nil, nil, startR\rangle$, $\langle nil, read, nil, startR\rangle$, $\langle stop, nil, nil, stopR\rangle$, $\langle nil, stop, nil, stopR\rangle$, $\langle nil, nil, write, startW\rangle$, $\langle nil, nil, stop, stopW\rangle$ and $\langle nil, nil, nil, nil\rangle$. A community behaves like an agent. Hence, we can import to the global level local reasoning techniques.

PROPOSITION 5. *Let* $q \in SC_{CSig}$*. If*

I1. $CSpec \vdash q_{[\vec{S}_0]}$;

I2. $CSpec \vdash \forall s, \vec{x}_1, \ldots, \vec{x}_n$
$(\vec{occurs}(\langle a_1(\vec{x}_1), \ldots, a_n(\vec{x}_n)\rangle, s) \wedge q_{[s]}) \rightarrow q_{[\vec{do}(\langle a_1(\vec{x}_1), \ldots, a_n(\vec{x}_n)\rangle, s)]}$,
for every $a_i \in Act^i$, $i \in ID$;

then $ASpec \vdash \forall s\ actual(s) \rightarrow q_{[s]}$.

Other results are also easily transposed to global level.

EXAMPLE 7. *One of the requirements of* $CSpec^{rw}$ *is that the* writer *should exclude all the* readers*. This is expressed by the state constraint:*

$$\forall s\ \vec{actual}(s) \rightarrow (3.holds(w?, [s_{\downarrow 3}]) \rightarrow$$
$$((\neg 1.holds(r?, [s_{\downarrow 1}])) \wedge (\neg 2.holds(r?, [s_{\downarrow 2}])))).$$

We start by proving two auxiliary results:

$$CSpec^{rw} \vdash \forall s\ \vec{actual}(s) \rightarrow (3.holds(w?, [s_{\downarrow 3}]) \leftrightarrow 4.holds(w?, [s_{\downarrow 4}])) \quad \text{(I)}$$

$$CSpec^{rw} \vdash \forall s\ \vec{actual}(s) \rightarrow (4.holds(nr(0), [s_{\downarrow 4}]) \leftrightarrow$$
$$(1.\neg holds(r?, [s_{\downarrow 1}]) \wedge 2.\neg holds(r?, [s_{\downarrow 2}]))) \quad \text{(II)}$$

(I) *establishes the relation between the writer and the buffer and* (II) *the relation between the readers and the buffer. The proof uses Proposition 5, following a strategy similar to Example 3. There, we proved:*

$$ASpec^4 \vdash \forall s\ actual(s) \rightarrow (holds(w?, [s]) \rightarrow holds(nr(0), [s])).$$

Using Definition 8, we can derive

$$CSpec^{rw} \vdash \forall s\ \vec{actual}(s) \rightarrow (4.holds(w?, [s_{\downarrow 4}]) \rightarrow 4.holds(nr(0), [s_{\downarrow 4}]))$$

which, together with (I) *and* (II)*, imply the desired result.*

Note that the previous example illustrates the case of a property that can only be specified and proved at global level.

4. Abduction

We introduce the notions of abduction problem in the context of an agent specification and of explanation for an abduction problem. Then we present techniques to obtain explanations for safety abduction problems. Afterwards, we extend them to agent community specifications.

4.1. Local abduction

Assume $ASpec = \langle ASig, AxF, AxS\rangle$ with $ASig = \langle Act, Att, \tau\rangle$ and consider $\tau(a) = \tau(f) = \epsilon$ for every $a \in Act$ and $f \in Att$. Suppose that some envisaged specification requirement is not inferred from $ASpec$. To ensure it, we can introduce further restrictions on the agent's behaviour, enriching $ASpec$ with new axioms. However, specification axioms are often restricted to formulae/schemata of a certain kind. If the formula representing the requirement is of a different kind, one cannot just add it to the specification. Suitable new axioms must be found. This task has an abductive nature [18] and we tackle this problem using abductive reasoning techniques. Following usual designations used therein, the formula representing the envisaged requirement is an *abduction problem* in *ASpec* and the new specification axioms constitute an *explanation* for the abduction problem.

DEFINITION 10. *A formula* $\varphi \in L_{\Sigma^{ASig}}$ *is an* abduction problem *in ASpec if* $ASpec \nvdash \varphi$. *An* explanation *for an abduction problem* $\varphi \in L_{\Sigma^{ASig}}$ *in ASpec is a finite set* $E \subseteq SC^{[S_0]}_{ASig} \cup EC_{ASig}$ *such that* $ASpec' = \langle ASig, AxF \cup E, AxS\rangle$ *is consistent and* $ASpec' \vdash \varphi$.

Our goal is to provide suitable restrictions on the agent's behaviour whenever a requirement is not fulfilled. Hence, our explanations consist of new initial conditions and enabling conditions (we do not consider changing the valuation schemata). We focus on the already considered safety properties. As expected, φ is a *safety abduction problem* in *ASpec* whenever φ is a safety formulae and is an abduction problem in *ASpec*. In the sequel, we present techniques for generating explanations for such problems. However, other safety properties, as well as other kinds of properties [11] (e.g., liveness properties) can be considered [8].

In general, there are several possible explanations for an abduction problem and, of course, some of them can be considered more interesting than others. Some preference criteria between explanations can be established. We can give preference to explanations that do not restrict the agent's behaviour more than it is necessary. However, depending on the problem in hand, one can consider other more suitable criteria.

EXAMPLE 8. *We consider a simplified version of the working example with only one reader. In this case the buffer can be simplified. It is enough to consider only one fluent* $r?$ *(signaling if the reader is in the critical section). We are going to drop some of the axioms and introduce a new action error. Then, the specification of the buffer is* $ASpec^B = \langle ASig^B, AxF^B, AxS^B\rangle$ *where*

– $ASig^B = \langle\{startR, startW, stopR, stopW, error\},\{r?, w?\}, \tau^b\rangle$ *where* $\tau^b(a) = \epsilon$, *for* $a \in \{startR, startW, stopR, stopW, error\}$, *and* $\tau^b(f) = \epsilon$ *for* $f \in \{r?, w?\}$.

– $AxF^B = \{\forall u\ poss(startR, u) \rightarrow \neg holds(w?, u),$
$\forall u\ poss(stopR, u) \rightarrow holds(r?, u),$
$\forall u\ poss(stopW, u) \rightarrow holds(w?, u),$
$\forall u\ poss(error, u) \rightarrow holds(w?, u)\ \}$;

– $AxS^B = \{\forall s\ (occurs(startR, s) \wedge \alpha_{[s]}{}^{r?}_{\boldsymbol{t}}) \rightarrow \alpha_{[do(startR,s)]},$
$\forall s\ (occurs(startW, s) \wedge \alpha_{[s]}{}^{w?}_{\boldsymbol{t}}) \rightarrow \alpha_{[do(startW,s)]},$
$\forall s\ (occurs(stopR, s) \wedge \alpha_{[s]}{}^{r?}_{\boldsymbol{f}}) \rightarrow \alpha_{[do(stopR,s)]},$
$\forall s\ (occurs(stopW, s) \wedge \alpha_{[s]}{}^{w?}_{\boldsymbol{f}}) \rightarrow \alpha_{[do(stopW,s)]},$
$\forall s\ (occurs(error, s) \wedge \alpha_{[s]}{}^{r?}_{\boldsymbol{t}}) \rightarrow \varphi_{[do(error,s)]}\}.$

Clearly, $\forall s\ actual(s) \rightarrow \neg(holds(r?, [s]) \wedge holds(w?, [s]))$ *is a safety abduction problem in* $ASpec^B$*.*

We dropped the initial conditions and the enabling conditions for $startW$. Thus, it is no more the case that the reader and the writer are prevented from accessing the critical section simultaneously. The formula stating the mutual exclusion is a safety problem in $ASpec^B$. We have to find an explanation for this problem, i.e., we have to find how to enrich $ASpec^B$ with suitable initial and enabling conditions. Hopefully, the proposed techniques for generation of explanations will recover the initial and enabling conditions considered before or, at least, the components of those conditions that are essential for ensuring mutual exclusion (besides this property, others must also be ensured, justifying eventual additional conditions in the former specification). Since we introduced the action *error*, clearly violating the envisaged property, generated explanations should also prevent its occurrence. The abductive techniques combine proof techniques for safety properties presented in [12] with tableau based abductive techniques [15, 8].

Consider a safety problem $\forall s, a occurs(a, s) \rightarrow (p_{[s]} \rightarrow q_{[do(a,s)]})$ and assume that, for every $a \in Act$ that does not establish q from p, we get an explanation for $\forall s\ (occurs(a, s) \wedge p_{[s]}) \rightarrow q_{[do(a,s)]}$. From Proposition 1, we easily conclude that, taking all those explanations, an explanation for the problem can be obtained. For safety problems $\forall s actual(s) \rightarrow q_{[s]}$, considering Proposition 2, we have to get explanations for the initial establishment of q and explanations for $\forall s(occurs(a, s) \wedge q_{[s]}) \rightarrow q_{[do(a,s)]}$ for each a that does not preserve q. Explanations for the initial establishment of q in $ASpec$ can be generated using a tableau for a set of formulae related to the ones in AxF and $\neg q$. This set is called the *generator of explanations for the initial establishment of* q. Similarly, explanations for the establishment of q from p by a in $ASpec$ can be generated using a tableau for a set of formulae related to formulae in AxF, schemata in AxS, $\neg q$ and p. This set is called the *generator of explanations for the establishment of* q *from* p *by* a. Each of these sets will be a finite subset of SC^u_{ASig} for some u. We just need propositional tableaux since atomic state constraints can be considered, for this purpose, propositional symbols. We consider tableaux as in [5] and the usual notions of *closed*, *open* and *exhausted tableau branch* and *closed*, *open* and *exhausted tableau*. There is a closed tableau for a set P of propositional formulae iff P is inconsistent. The notion of tableau *closure* is also relevant. Explanations are generated from tableau closures.

DEFINITION 11. *Let t be an exhausted open tableau for a finite subset of SC^u_{ASig}. A* closure *of t is a non empty set Cl of literals such that: (i) for each open branch ρ of t there is a literal l in ρ such that its conjugate, $\bar{l}$, is in Cl, (ii) $flt(l), flt(\bar{l}) \notin \{\boldsymbol{t}, \boldsymbol{f}\}$ for every $l \in Cl$ and (iii) for every literal l, if $l \in Cl$ then $\bar{l} \notin Cl$. A closure Cl of t is said to be* minimal *if there is no other closure Cl' of t such that $Cl' \subset Cl$.*

A closure of a tableau t for a set P is a nonempty set of literals, not involving **t** or **f**, that does not contain conjugate literals, and including the conjugate of a literal of each branch of t. If t is open and Cl is one of its closures, then there is a closed tableau for $P \cup Cl$.

EXAMPLE 9. *Consider the specification $ASpec^B$ in Example 8. Figure 1 depicts a tableau for $\{holds(r?,u) \wedge holds(\boldsymbol{t},u), \neg(holds(r?,u) \wedge holds(w?,u))\} \subseteq SC^u_{ASig}$. The tableau is open and exhausted.*

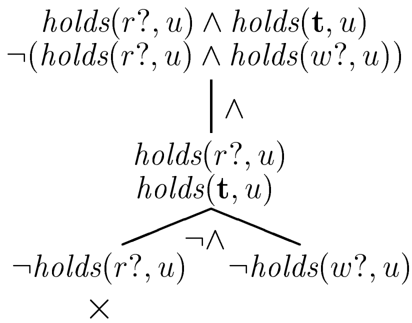

Figure 1. Tableau for $\{holds(r?,u) \wedge holds(\mathbf{t},u), \neg(holds(r?,u) \wedge holds(w?,u))\}$

The sets $Cl_1 = \{\neg holds(r?,u)\}$ and $Cl_2 = \{holds(w?,u)\}$ are closures of the tableau. They are minimal closures.

PROPOSITION 6. *Let t be an exhausted open tableau for a set P of propositional formulae. If Cl_i, $1 \leq i \leq n$, are closures of t then there is a closed tableau for $P \cup \{\bigvee_{1 \leq i \leq n}(\bigwedge Cl_i)\}$. Hence, this set is inconsistent.*

Proof. Assuming the usual tableau rule $\vee$ ($\wedge$) generalized to disjunctions (conjunctions) of more than two formulae, a closed tableau t' for the set can be obtained from t. For each open branch ρ of t: (i) apply rule $\vee$ to $\bigvee_{1\leq i\leq n}(\bigwedge Cl_i)$ obtaining new branches ρ_i^1, $1 \leq i \leq n$, each ρ_i^1 extending ρ with a node labeled with $\bigwedge Cl_i$; (ii) for each ρ_i^1, $1 \leq i \leq n$, apply rule $\wedge$ to $\bigwedge Cl_i$ obtaining the branch ρ_i^2 which extends ρ_i^1 with a node labeled with the literals in Cl_i. By Definition 11, t' is closed.

Techniques for generating explanations for the initial establishment of q and the establishment of q from p by a are based on the next result.

PROPOSITION 7. *Let* $AxS \cap S^a_{ASig} = \{\forall s\ (occurs(a,s) \wedge \alpha_{[s]p_1\ldots p_n}^{\ \ f_1\ldots f_n}) \rightarrow \alpha_{[do(a,s)]}\}$ *for* $a \in Act$*, and let* u *be a variable of sort stt and* $p, q \in SC_{ASig}$*.*

1. *The generator of explanations for the initial establishment of* q *in ASpec is the set*

$$G^{\overset{i}{\rightarrow} q}_{ASpec} = (AxF \cap SC^{[S_0]}_{ASig}) \cup \{\neg q_{[S_0]}\}.$$

If there is a closed tableau for $G^{\overset{i}{\rightarrow} q}_{ASpec}$ *then* q *is initially established.*

2. *The generator of explanations for the establishment of* q *from* p *(at* u*) by* $a \in Act$ *in ASpec is the set*

$$G^{p \overset{a}{\rightarrow} q}_{ASpec} = cnd(AxF \cap EC^a_{ASig})_u \cup \{p_u, \neg q_{up_1\ldots p_n}^{\ f_1\ldots f_n}\}.$$

If there is a closed tableau for $G^{p \overset{a}{\rightarrow} q}_{ASpec}$ *then* a *establishes* q *from* p*.*

Proof. (Sketch)

1. Consider a first-order interpretation structure M satisfying Ax_{SSC} and *ASpec*. M does not satisfy $G^{\overset{i}{\rightarrow} q}_{ASpec}$ (the set is inconsistent). Since M satisfies $AxF \cap SC^{[S_0]}_{ASig}$, M does not satisfy $\neg q_{[S_0]}$ thus M satisfies $q_{[S_0]}$. Hence, $q_{[S_0]}$ is a semantic consequence (in the usual sense) of Ax_{SSC} and *ASpec*. Using completeness, q is initially established in *ASpec*.

2. Consider a first-order interpretation structure M satisfying Ax_{SSC}, *ASpec*, $occurs(a,s)$ and p_u, such that $[s] = u$ (for each u, it is always possible to find such an s, due to axiom 2.2). As a consequence, M must satisfy $poss(a,u)$ and so M also satisfies $cnd(AxF \cap EC^a_{ASig})_u$. Since M does not satisfy $G^{p \overset{a}{\rightarrow} q}_{ASpec}$, M must satisfy $q_{up_1\ldots p_n}^{\ f_1\ldots f_n}$. M satisfies the valuation schema for a instantiated with $\alpha = q$, thus M satisfies $q_{[do(a,s)]}$. Hence, $\forall(occurs(a,s) \rightarrow (p_{[s]}) \rightarrow q_{[do(a,s)]})$ is a semantic consequence of Ax_{SSC} and *ASpec*. Using completeness, a establishes q from p in *ASpec*.

PROPOSITION 8. *Let* $a \in Act$ *and* $p, q \in SC_{ASig}$*.*

1. *If* $Cl_i, \ldots, Cl_n$ *are closures of an exhausted open tableau for the set* $G^{\overset{i}{\rightarrow} q}_{ASpec}$ *and* $\overline{p} = \bigvee_{1\leq i\leq n}(\bigwedge Cl_i)$ *then* q *is initially established in* $ASpec' = \langle ASig, AxF \cup \{\overline{p}\}, AxS\rangle$*.*

2. *If $Cl_1, \ldots, Cl_n$ are closures of an exhausted open tableau for the set $G^{p \xrightarrow{a} q}_{ASpec}$ and $\overline{\varphi} = \forall u \; poss(a,u) \rightarrow (\bigvee_{1 \leq i \leq n}(\bigwedge Cl_i))_u$ then a establishes q from p in $ASpec' = \langle \Sigma, AxF \cup \{\overline{\varphi}\}, AxS \rangle$.*

Proof. These results easily follow from Propositions 6 and 7.

We can get an explanation for the initial establishment of q as follows: take one or more closures of an open tableau for $G^{\xrightarrow{i} q}_{ASpec}$, get $\overline{p}$ from these closures as described above and $\{\overline{p}\}$ constitutes an explanation whenever $ASpec' = \langle ASig, AxF \cup \{\overline{p}\}, AxS \rangle$ is consistent. Similarly, from closures of an open tableau for $G^{p \xrightarrow{a} q}_{ASpec}$ we conclude that $\{\overline{\varphi}\}$ constitutes an explanation for the establishment of q from p by a whenever $ASpec' = \langle \Sigma, AxF \cup \{\overline{\varphi}\}, AxS \rangle$ is consistent. In the sequel, we often identify a singular explanation with the formula it contains.

Given state constraints $\overline{p}$ and $\overline{p}'$ obtained as in Proposition 8, we say that $\overline{p}$ is more *liberal* than $\overline{p}'$ iff $Ax_{SSC} \vdash \overline{p}' \rightarrow \overline{p}$ and $Ax_{SSC} \not\vdash \overline{p} \rightarrow \overline{p}'$. Given enabling conditions $\overline{\varphi}$ and $\overline{\varphi}'$ obtained as in Proposition 8, $\overline{\varphi}$ is more *liberal* than $\overline{\varphi}'$ iff $Ax_{SSC} \vdash cnd(\overline{\varphi}') \rightarrow cnd(\overline{\varphi})$ and $Ax_{SSC} \not\vdash cnd(\overline{\varphi}) \rightarrow cnd(\overline{\varphi}')$. This notion can be used as preference criterion between explanations: more liberal explanations restrict less the agent's behaviour. The more liberal explanations can be obtained from disjunctions of minimal closures (see [7]).

EXAMPLE 10. *Consider $ASpec^B$ as in Example 8. The state constraint $q = \neg(holds(r?,u) \wedge holds(w?,u))$ is not initially established. The generator of explanations for the initial establishment of q is $\{\neg q_{[S_0]}\}$:*

$$G^{\xrightarrow{i} q}_{ASpec^B} = \{\neg\neg(holds(r?,[S_0]) \wedge holds(w?,[S_0]))\}.$$

Figure 2 depicts an open and exhausted tableau t for $G^{\xrightarrow{i} q}_{ASpec^B}$. The sets $Cl_1 = \{\neg holds(r?,[S_0])\}$ and $Cl_2 = \{\neg holds(w?,[S_0])\}$ are closures of t. We easily conclude that the following formulae are explanations for the initial establishment of q:

$$\neg holds(r?,[S_0]) \tag{10.1}$$

$$\neg holds(w?,[S_0]) \tag{10.2}$$

$$(\neg holds(w?,[S_0])) \vee (\neg holds(r?,[S_0])). \tag{10.3}$$

Clearly, 10.3 is more liberal than 10.1 and 10.2. Considering $ASpec^B$ in Example 2, and taking into account the simplified version we are working with herein, one could expect that the generated explanations would recover two initial conditions ($\neg holds(r?,[S_0])$ and $\neg holds(w?,[S_0])$),

$$\neg\neg(holds(r?, [S_0]) \wedge holds(w?, [S_0]))$$
$$\Big|\ \neg\neg$$
$$holds(r?, [S_0]) \wedge holds(w?, [S_0])$$
$$\Big|\ \wedge$$
$$holds(r?, [S_0])$$
$$holds(w?, [S_0])$$

Figure 2. Tableau for $\{\neg\neg(holds(r?, [S_0]) \wedge holds(w?, [S_0]))\}$

similar to ones ones we had. However, note that to ensure $q_{[S_0]}$ *we only need one of them. Note also that* $\{\neg holds(r?, [S_0]), \neg holds(w?, [S_0])\}$ *is a closure of* t *(although not a minimal one). Using the technique described above, we could also generate an explanation involving both conditions.*

EXAMPLE 11. *Consider again Example 8. Clearly, startW does not preserve* $q = \neg(holds(r?, u) \wedge holds(w?, u))$*. The set generator of explanations for the preservation of q by startW is* $\{q_u, \neg {q_u}^{w?}_{\boldsymbol{t}}\}$*, i.e.,*

$$G^{q \overset{startW}{\to} q}_{ASpec^B} = \{\neg(holds(r?, u) \wedge holds(w?, u)), \neg\neg holds(r?, u) \wedge holds(\boldsymbol{t}, u)\}$$

and it is easy to conclude that every exhausted tableau for this set is open and the closures are exactly the closures of the tableau in Example 9. Recalling Cl_1 *and* Cl_2 *presented therein, we easily conclude that*

$$\forall u\ poss(startW, u) \rightarrow (\neg holds(r?, u)) \tag{11.1}$$

$$\forall u\ poss(startW, u) \rightarrow holds(w?, u) \tag{11.2}$$

$$\forall u\ poss(startW, u) \rightarrow (\neg holds(r?, u) \vee holds(w?, u)) \tag{11.3}$$

are explanations for the preservation of q by startW. Enabling condition 11.1 only allows the occurrence of startW when r? does not hold. Since startW does not change r?, adding this axiom ensures that r? also does not hold (hence, q holds) after startW. This condition only allows the occurrence of startW when q also holds ($Ax_{SSC} \vdash (\neg holds(r?, u)) \rightarrow q$)*.*

Consider 11.2 and recall that the goal, for the moment, is the preservation *of q by startW, that is, we only have to ensure q after startW when this action occurs when q holds. So, when q does not hold (i.e., both r? and w? hold), startW should be free to occur and, indeed, 11.2 allows the occurrence of startW. Assume now that q holds. 11.2 ensures that startW only occurs when w? also holds. Obviously, if both q and*

$w?$ hold then $r?$ does not hold. Since $startW$ does not change $r?$, q holds after $startW$. So, this condition allows the occurrence of $startW$ when q does not hold and in some of the situations where q holds.

Explanation 11.3 is more liberal than 11.1 and 11.2. It imposes less restrictions on the agent's behaviour. Indeed, if neither $w?$ nor $r?$ hold, $startW$ can occur (q is preserved), but adding 11.2 prevents the occurrence of $startW$ in this case. On the other hand, if both $w?$ and $r?$ hold, again $startW$ can occur, but adding 11.1 prevents it. However, in both cases, the explanation 11.3 allows the occurrence of $startW$.

Note that none of the previous explanations is the condition in Example 2 (which in the simplified version of the working example should be $\forall u\ poss(startW, u) \rightarrow (\neg holds(r?, u) \wedge (\neg holds(w?, u)))$). However, the conjunct $\neg holds(w?, u)$ is irrelevant for ensuring the preservation of q. Below we will comment again on these explanations in the context of a safety problem involving q.

It may happen that no closure for an exhausted tableau for $G^{p \xrightarrow{a} q}_{ASpec}$ exists. In this cases a establishes $\neg q$ (see [7]). Thus, the only way to restrict the occurrence of a in order to establish q from p is to forbid its occurrence. In this case $\forall u\ poss(a, u) \rightarrow holds(\mathbf{f}, u)$ can provide an explanation. Another interesting remark is the following. Assume that $\forall u poss(a, u) \rightarrow holds(f, u) \in AxF$. Then, $\forall u\ poss(a, u) \rightarrow (\neg holds(f, u))$ could be considered an explanation for establishing q from p by a. In fact, in the enriched specification, the occurrence of a is not possible since a would only be enabled whenever f and $\neg f$ both held. Thus, a trivially establishes q from p, for all p and q. Trivial explanations similar to this one do not take into account the problem in hand. We may choose to avoid them. Proposition 9 identifies tableau closures that generate such explanations. Using only tableau closures other than these prevents, whenever wanted, the generation of such explanations.

PROPOSITION 9. Let $a \in Act$. Assume that $Cl_1, \ldots, Cl_n$, $n \geq 1$, are closures both of an open exhausted tableau for $G^{p \xrightarrow{a} q}_{ASpec}$ and of an open exhausted tableau for $cnd(AxF \cap EC^a_{ASig})_u \subseteq G^{p \xrightarrow{a} q}_{ASpec}$. Let $\overline{\varphi} = \forall u\ poss(a, u) \rightarrow \bigvee_{1 \leq i \leq n} (\bigwedge Cl_i)_u$. We have that a establishes q' from p' in $ASpec' = (\Sigma, AxF \cup \{\overline{\varphi}\}, AxS)$ for any $p', q' \in SC_{ASig}$.

Proof. : Clearly $cnd(AxF \cap EC^a_{ASig})_u \cup \{\bigvee_{1 \leq i \leq n} (\bigwedge Cl_i)_u\}$ is inconsistent (Prop. 6). Hence, $ASpec'$ never allows the occurrence of a.

Another question is the following: do the closures of tableaux for $G^{\xrightarrow{i} q}_{ASpec}$ and $G^{p \xrightarrow{a} q}_{ASpec}$ depend on the particular open exhausted tableau for them we consider? The answer is negative: closures are independent on the particular tableau, i.e., if t and t' are open exhausted tableaux for the same set of propositional formulae then they have that same set of closures [7]. The following proposition summarizes the methods for generating explanations for the safety problems described above.

PROPOSITION 10. For each $p, q \in SC_{ASig}$, let A_{pq} be the set of actions that do not establish q from p in *ASpec*. For each $a \in A_{pq}$, let E_a be an explanation for the establishment of q from p by a in *ASpec*. Let E_q^I be an explanation for the initial establishment of q in *ASpec*.

1. The set $E = \bigcup_{a \in A_{pq}} E_a$ is an explanation for the safety problem $\forall s, a\ occurs(a,s) \rightarrow (p_{[s]} \rightarrow q_{[do(a,s)]})$ in *ASpec* whenever $ASpec' = \langle ASig, AxF \cup E, AxS \rangle$ is consistent.

2. The set $E = E_q^I \cup (\bigcup_{a \in A_{qq}} E_a)$ is an explanation for the safety problem $\forall s\ actual(s) \rightarrow q_{[s]}$ in *ASpec* whenever $ASpec' = \langle ASig, AxF \cup E, AxS \rangle$ is consistent.

Proof. The results easily follow from Propositions 1, 2 and 8.

EXAMPLE 12. *Recall* $ASpec^B$ *in Example 8,* $q = \neg(holds(r?, u) \wedge holds(w?, u))$ *and the safety abduction problem* $\forall s\ actual(s) \rightarrow q_{[s]}$.

1. The state constraint q *is not initially established. In Example 10 we have generated explanations for the initial establishment of* q.

2. It is easy to conclude that the action startR preserves q*: consider Proposition 7 and the fact that there is a closed tableau for*

$$\begin{aligned} G_{ASpec^B}^{q \overset{startR}{\rightarrow} q} &= \{\neg holds(w?, u), \neg(holds(r?, u) \wedge holds(w?, u)), \\ &\quad \neg\neg(holds(\boldsymbol{t}, u) \wedge holds(w?, u))\}. \end{aligned}$$

Reasoning as above we conclude that stopR and nil also preserve q.

3. The action startW does not preserve q*. In Example 11 we have generated explanations for the preservation of* q *by startW.*

4. The action error does not preserve q*. The generator of explanations for the preservation of* q *by error is*

$$\begin{aligned} G_{ASpec^B}^{q \overset{error}{\rightarrow} q} &= \{holds(w?, u), \neg(holds(r?, u) \wedge holds(w?, u)), \\ &\quad \neg\neg(holds(\boldsymbol{t}, u) \wedge holds(w?, u))\}. \end{aligned}$$

The minimal closures of any exhausted tableau for this set are $Cl_1 = \{\neg holds(w?, u)\}$ *and* $Cl_2 = \{holds(r?, u)\}$*. Clearly, the following are explanations for the preservation of* q *by error:*

$$\forall u\ poss(error, u) \rightarrow (\neg holds(w?, u))$$

$$\forall u\, poss(error, u) \rightarrow holds(r?, u).$$

6. We easily conclude that the following are examples of explanations for the abduction problem in hand:

$$\begin{aligned}
E_1 &= \{\neg holds(w?, [S_0]), \forall u\; poss(startW, u) \rightarrow (\neg holds(r?, u)),\\
&\quad \forall u\; poss(error, u) \rightarrow (\neg holds(w?, u))\}\\
E_2 &= \{\neg holds(w?, [S_0]), \forall u\; poss(startW, u) \rightarrow holds(w?, u),\\
&\quad \forall u\; poss(error, u) \rightarrow holds(r?, u)\}\\
E_3 &= \{\neg holds(w?, [S_0]), \forall u\; poss(error, u) \rightarrow (\neg holds(w?, u))\\
&\quad \forall u\; poss(startW, u) \rightarrow (\neg holds(r?, u) \vee holds(w?, u))\}.
\end{aligned}$$

A few remarks are in order. Consider E_2 and the condition therein for startW. In Example 11, we noted that this condition allows the occurrence of startW when, in particular, q does not hold. Since our goal, herein, is to ensure q in the state of every actual situation, it is irrelevant, for this purpose, that startW is enabled when q does not hold. Since if both q and w? hold, then r? does not hold, we can say that for the purpose in hand condition in E_2 is somehow equivalent to $\forall u\; poss(startW, u) \rightarrow (\neg holds(r?, u))$ in E_1 (both restrict the occurrence of startW to situations where r? does not hold). The condition in E_1 makes this fact explicit while this is implicit in the other condition. For this reason, between these to explanations preference could be given to $\forall u\; poss(startW, u) \rightarrow (\neg holds(r?, u))$. Similar remarks can be made about the enabling conditions for action error.

In Example 11 we noted that the condition for startW in E_3 is more liberal than the others. However, recalling again that the purpose herein is to ensure the invariance of q, ensuring that startW only occurs provided that $(\neg holds(r?, u) \vee holds(w?, u))$ or ensuring that it only occurs provided that $\neg holds(r?, u)$ is, herein, the same (since if q and w? holds then r? does not hold). The information that startW is enabled when w? holds, is in this case superfluous. Thus, $\forall u\; poss(startW, u) \rightarrow (\neg holds(r?, u))$ is more interesting since it is (syntactically) simpler and expresses the same restrictions on startW.

From above, we conclude that E_1 is more interesting than the others. Note that E_1 complies with the remarks made after Example 8: the new axioms in E_1 recover the conditions we have dropped (in fact, the components of the conditions that are relevant for ensuring the property considered herein) and prevent the occurrence of error as expected.

The remarks in Example 12 on preference criteria for explanations when dealing with a safety problem involving the invariance of a state constraint can be generalized to the case of any safety problem of this kind. We can detect the unwanted explanations and ensure only the generation of the more simple or explicit ones. The idea is to discard a minimal closure Cl of the tableau for $G^{q \xrightarrow{a} q}_{ASpec}$ if there is another minimal closure Cl' of it such that $Ax_{SSC} \vdash ((\bigwedge Cl) \wedge q) \rightarrow (\bigwedge Cl')$.

4.2. Global abduction

We extend the abductive techniques developed for agent specifications to abduction problems in community specifications. Abduction problems can now involve properties encompassing the behaviour of several agents. In what concerns explanations for a given abduction problem, we can now consider a new possibility of constraining the community behaviour: we can also specify new interactions among agents. Obviously, this can only be achieved working at community level.

Explanations are now families of sets of formulae where each set contains new specification axioms for an agent specification or new community interaction axioms. We assume $CSig = \langle ID, Sig \rangle$ and $CSpec = \langle CSig, AxF, AxS \rangle$, again considering $\tau(a) = \tau(f) = \epsilon$.

DEFINITION 12. *A formula* $\varphi \in L_{\Sigma CSig}$ *is an* abduction problem *in CSpec if* $CSpec \not\vdash \varphi$. *An* explanation *for an abduction problem* $\varphi \in L_{\Sigma CSig}$ *in CSpec is a family of sets of formulae* $\mathcal{E} = \{E_i\}_{i \in ID \cup \{0\}}$ *such that*

- $E_i \subseteq SC_{Sig(i)} \cup EC_{Sig(i)}$ *for each* $i \in ID$ *and* $E_0 \subseteq IL_{CSig}$,
- $CSpec' = (CSig, \{AxF_i \cup E_i\}_{i \in ID \cup \{0\}}, \{AxS_i\}_{i \in ID})$ *is consistent*
- $CSpec' \vdash \varphi$.

We concentrate on safety abuction problems $\forall s\ \vec{actual}(s) \rightarrow q_{[s]}$ or $\forall s, a\ \vec{occurs}(a, s) \rightarrow (p_{[s]} \rightarrow q_{[\vec{do}(a,s)]})$, for some $p, q \in SC_{CSig}$, as before.

EXAMPLE 13. *Recall the simplified version of the working example (only one reader). The community signature is* $CSig^{rw} = \langle \{1, 2, 3\}, Sig^{rw} \rangle$ *where* $Sig^{rw}(1) = ASig^R$, $Sig^{rw}(2) = ASig^W$ *and* $Sig^{rw}(3) = ASig^B$ *as in Example 2. Considering* AxS^B *of Example 8 and the agent specifications* $ASpec^R$, $ASpec^W$ *of Example 2, the community specification* $CSpec^{rw} = \langle CSig^{rw}, AxF^{rw}, AxS^{rw} \rangle$ *is as follows:*

- $AxF^{rw}_1 = AxF^R$, $AxF^{rw}_2 = AxF^W$

$$\begin{aligned} AxF^{rw}_3 = \{ & \forall u\ poss(startR, u) \rightarrow \neg holds(w?, u), \\ & \forall u\ poss(startW, u) \rightarrow (\neg holds(r?, u) \wedge \neg holds(w?, u)), \\ & \forall u\ poss(stopR, u) \rightarrow holds(r?, u), \\ & \forall u\ poss(stopW, u) \rightarrow holds(w?, u); \end{aligned}$$

– $AxF_0^{rw} = \{1.do(read, s) \rhd 3.do(startR, s),$
$1.do(stop, s) \rhd 3.do(stopR, s),$
$3.do(stopR, s) \rhd 1.do(stop, s),$
$2.do(write, s) \rhd 3.do(startW, s),$
$3.do(startW, s) \rhd 2.do(write, s),$
$2.do(stop, s) \rhd 3.do(stopW, s),$
$3.do(stopW, s) \rhd 2.do(stop, s)\ \}$

– $AxS_1^{rw} = AxF^R,\ AxS_2^{rw} = AxS^W$
$AxS_3^{rw} = AxS^B \backslash \{\forall s\ (occurs(error, s) \wedge \alpha_{[s]}{}^{r?}_{t}) \rightarrow \varphi_{[do(error,s)]}\}$;

We have that $\forall s\ act\vec{u}al(s) \rightarrow (1.holds(r?, [s_{\downarrow 1}]) \leftrightarrow 3.holds(r?, [s_{\downarrow 3}]))$ *is a safety abduction problem in* $CSpec^{rw}$.

We have dropped the initial conditions for the *buffer* and one of the interaction axioms, fundamental for ensuring synchronization between the *reader* and the *buffer*. The proposed abductive techniques should be able to recover these axioms (or relevant similar ones). The techniques for the generation of explanations are similar to the ones presented at local level. However, when establishing one state constraint from another we have to take into account that we now have global actions.

To get explanations for the initial establishment of q, we consider an exhausted tableau for a set of formulae in SC_{CSig} that takes now into account the initial conditions of the agents in $ID(q)$. An explanation is a family of sets of state constraints in the language of the agents in $ID(q)$ expressing new initial conditions for them. The state constraints are obtained from closures of tableaux. Since each closure is a subset of SC_{CSig}, we have to extract from it the literals $i.l$ related to each agent $i \in ID(q)$, that are then translated back to the language of i. A new state constraint for each agent in $ID(q)$ can then be constructed.

To get explanations for establishing q from p by a global action a, we consider an exhausted tableau for a set of formulae that takes into account the enabling constraints and the valuation schema specified for $a_{\downarrow i}$, for each $i \in ID(\{p, q\})$, but also the specified interactions between the agents involved. Explanations include new enabling conditions (for agents in $ID(\{p, q\})$) but we also consider new interactions (for AxF_0). The generation of explanations is based on the following results.

PROPOSITION 11. *Let* $\vec{a} = \langle a_1, \ldots, a_n \rangle$, *where* $a_i \in Act^i$ *for each* $i \in ID$, $p, q \in SC_{CSig}$ *and* u *a variable of sort stt. For each* $a_i \in Act^i$ *and* $i \in ID$, *let* $AxS_i \cap S^{a_i}_{Sig(i)} = \{\forall s\ (occurs(a_i, s) \wedge \alpha_{[s]p_1^i \ldots p_{n_i}^i}^{\;f_1^i \ldots f_{n_i}^i}) \rightarrow \alpha_{[do(a_i,s)]}\}$.

1. *The generator of explanations for the initial establishment of q in CSpec is the set*

$$G_{CSpec}^{\stackrel{i}{\to} q} = (\bigcup_{j \in ID(q)} j.(AxF_j \cap SC_{Sig(j)}^{[S_0]})) \cup \{\neg q_{[\vec{S_0}]}\}.$$

If there is a closed tableau for $G_{CSpec}^{\stackrel{i}{\to} q}$ then q is initially established.

2. *The generator of explanations for the establishment of q from p (at u) by $\vec{a}$ in CSpec is the set*

$$G_{CSpec}^{p \stackrel{\vec{a}}{\to} q} = (\bigcup_{i \in ID(\{p,q\})} i.(cnd(AxF_i \cap EC_{Sig(i)}^{a_i})_u))$$

$$\cup$$

$$\{\neg cnd(\varphi) : \varphi \in AxF_0, tri(\varphi) = i.a_i \textit{ for some } i \in ID(\{p,q\}) \textit{ and } j.a_j \notin tar(\varphi) \textit{ for all } j \in ID\}_u$$

$$\cup$$

$$\{p, \neg q_{1.p_1^1 \ldots 1.p_{n_1}^1 \ldots n.p_1^n \ldots n.p_{n_n}^n}^{1.f_1^1 \ldots 1.f_{n_1}^1 \ldots n.f_1^n \ldots n.f_{n_n}^n}\}_u.$$

If there is a closed tableau for $G_{CSpec}^{p \stackrel{\vec{a}}{\to} q}$ then $\vec{a}$ establishes q from p.

Proof. Similar to the proof of Proposition 7 taking into account that we also have to consider the preconditions of interaction formulae.

The set $G_{CSpec}^{\stackrel{i}{\to} q}$ includes the instantiation of q with $[\vec{S_0}]$ and initial conditions of agents whose attributes are involved in q. The set $G_{CSpec}^{p \stackrel{\vec{a}}{\to} q}$ includes the conditions of the enablings for actions in $\vec{a} = \langle a_1, \ldots, a_n \rangle$ belonging to the agents involved in p or q. Note that $\vec{a}$ cannot occur if $i.q' \to (i.a_i \rhd j.b_j) \in AxF_0$, $i.q'$ holds and $a_j \neq b_j$ (thus $\vec{a}$ trivially establishes q from p in this case). As a consequence, we only have to worry when $i.q'$ does not hold. This is the reason why we also introduce the negation of the conditions of the interaction formulae whose trigger is in $\vec{a}$ but whose targets do not belong to $\vec{a}$. As expected, the set also includes formulae related to the problem in hand. In $G_{CSpec}^{\stackrel{i}{\to} q}$ and $G_{CSpec}^{p \stackrel{\vec{a}}{\to} q}$ we only have to consider formulae of the agents involved in p or q and we are only going to generate explanations for those agents. This is an advantage of our modular approach to system specification.

PROPOSITION 12. *Let $\vec{a} = \langle a_1, \ldots, a_n \rangle$, where $a_i \in Act^i$ for each $i \in ID$, and $p, q \in SC_{CSig}$.*

1. *Let Cl be a closure of an exhausted tableau for $G^{\overset{i}{\to} q}_{CSpec}$. Consider the family of sets $\mathcal{E} = \{E_j\}_{j \in ID \cup \{0\}}$ such that $E_j = \{\bigwedge Cl_{\downarrow j}\}$ if $Cl_{\downarrow j} \neq \emptyset$ and $j \in ID$, and $E_j = \emptyset$ otherwise. We have that q is initially established in $CSpec' = \langle CSig, \{AxF_j \cup E_j\}_{j \in ID \cup \{0\}}, \{AxS_j\}_{j \in ID}\rangle$.*

2. *Let Cl be a closure of an exhausted tableau for $G^{p \overset{\vec{a}}{\to} q}_{CSpec}$ such that $a_i \neq nil$ for all $i \in ID(Cl)$.*

   (i) Let $j, j' \in ID(\{p, q\})$ with $j \neq j'$. Consider the family of sets $\mathcal{E} = \{E_i\}_{i \in ID \cup \{0\}}$ such that $E_i = \emptyset$ if $i \in ID$ and

   $$E_0 = \{j.do(a_j, s) \triangleright j'.do(b_{j'}, s)\}$$

   where $b_{j'} \in Act^{j'} \backslash \{a_{j'}, nil\}$. We have that $\vec{a}$ establishes q from p in $CSpec' = \langle CSig, \{AxF_i \cup E_i\}_{i \in ID \cup \{0\}}, \{AxS_i\}_{i \in ID}\rangle$.

   (ii) Consider the family of sets $\mathcal{E} = \{E_i\}_{i \in ID \cup \{0\}}$ such that $E_i = \emptyset$ for each $i \notin ID(Cl) \cup \{0\}$ and, for each $i \in ID(Cl)$ one of the following conditions hold

   - *$E_i = \{\forall u \; poss(a_i, u) \to q'\}$, where $q' = \bigwedge Cl_{\downarrow i}$ or $q' = holds(\boldsymbol{f}, u)$, and $tri(\varphi) \neq i.a_i$ for all $\varphi \in E_0$*
   - *$E_i = \emptyset$ and $\neg((\bigwedge Cl_{\downarrow i}))_{[s]} \to (i.do(a_i, s) \triangleright j.do(b_j, s)) \in E_0$ where $j \in ID(\{p, q\}) \backslash \{i\}$ and $b_j \in Act^j \backslash \{a_j, nil\}$.*

   We have that $\vec{a}$ establishes q from p in $CSpec' = \langle CSig, \{AxF_i \cup E_i\}_{i \in ID \cup \{0\}}, \{AxS_i\}_{i \in ID}\rangle$.

Proof. Similar to the proof of Proposition 8 taking into account that we also have to consider the preconditions of interaction formulae.

We get explanations for the initial establishment of q extracting from a closure of $G^{\overset{i}{\to} q}_{CSpec}$ the set of literals corresponding to each agent involved in q. From each non empty set of literals we obtain a new initial condition for the agent. For establishing q from p by $\vec{a}$, we can get a trivial explanation considering just an unconditional interaction whose trigger is an action in $\vec{a}$ and whose target is not an action in $\vec{a}$. This interaction is an explanation because $\vec{a}$ will never be enabled in the enriched specification. We can also obtain explanations from closures of $G^{p \overset{\vec{a}}{\to} q}_{CSpec}$. From such a closure and for each agent involved in p or q, we extract the corresponding set of literals. From each non empty set of literals we obtain either a new enabling condition (for the action a_i of the corresponding agent in $\vec{a}$) or a conditional interaction (whose trigger is a_i and whose target is not an action in $\vec{a}$), but not both. Naturally, *nil* is never involved in these explanations.

To get explanations for safety problems we proceed as at local level: whenever needed, we get explanations for the initial establishment of state constraints and for establishing one state constraint from another by a global action. Preference criteria among explanations could also be considered as discussed for local level.

EXAMPLE 14. *Let* $q = 1.holds(r?, u) \leftrightarrow 3.holds(r?, u)$.

1. The state constraint q *is not initially established. The generator of the initial establishment of* q *is*

$G^{\stackrel{i}{\rightarrow} q}_{Cspec^{rw}} = \{1.\neg holds(r?, [\vec{S_0}]), \neg(1.holds(r?, [\vec{S_0}]) \leftrightarrow 3.holds(r?, [\vec{S_0}]))\}$.

From an open exhausted tableau for this set we get the minimal closures $Cl = \{1.holds(r?, [\vec{S_0}])\}$ *and* $Cl' = \{3.\neg holds(r?, [\vec{S_0}])\}$. *Clearly, including* $holds(r?, [\vec{S_0}])$ *in* AxF_1 *will lead to an inconsistency. The suitable explanation is* $\{E_i\}_{i\in\{0,1,2,3\}}$ *where* $E_0 = E_1 = E_2 = \emptyset$ *and* $E_3 = \{\neg holds(r?, [\vec{S_0}])\}$.

2. Considering the interactions in AxF_0, *the relevant global actions are:* $\langle nil, nil, nil\rangle$, $\langle nil, stop, stopW\rangle$, $\langle nil, write, startW\rangle$, $\langle stop, nil, stopR\rangle$ $\langle read, nil, startR\rangle$ *and* $\langle nil, nil, startR\rangle$. *The first three of them preserve* q *because the actions involved do not change the attributes in* q. *The next two also preserve* q *because they change the attributes in* q *in the same way. The global action* $\vec{a} = \langle nil, nil, startR\rangle$ *does not preserve* q. *The generator of explanations for the preservation of* q *by* $\vec{a}$ *is*

$$\begin{aligned} G^{q \stackrel{\vec{a}}{\rightarrow} q}_{CSpec^{rw}} &= \{1.\neg holds(w?, u), 1.holds(r?, u) \leftrightarrow 3.holds(r?, u), \\ &\quad \neg(1.holds(r?, u) \leftrightarrow 3.holds(\boldsymbol{t}, u))\}. \end{aligned}$$

From an open exhausted tableau for this set we get the minimal closures $Cl = \{1.holds(r?, u)\}$, $Cl' = \{1.holds(w?, u)\}$ *and* $Cl'' = \{3.holds(r?, u)\}$. *The first two are not interesting, since the action from agent 1 in* $\vec{a}$ *is* nil. *From* Cl'' *we get* $\{E^1_i\}_{i\in\{0,1,2,3\}}$ *where* $E^1_1 = E^1_2 = E^1_3 = \emptyset$ *and* $E^1_0 = \{3.\neg holds(r?, [s]) \rightarrow (3.do(startR, s) \triangleright 1.do(read, s))\}$. *Note that whenever* $r?$ *holds in agent 3 (buffer), and assuming that* q *holds, we have that* $r?$ *also holds in agent 1 (reader). So, startR can occur in the buffer at the same time as nil occurs in the reader, since* $r?$ *holds in agent 3 after the occurrence of startR, ensuring the preservation of* q. *Thus, interaction with read is only needed when* $r?$ *does not hold in the buffer. If we really want to allow this kind of interactions in this case is another matter, but they indeed ensure the preservation of* q.

Disregarding the closures, the unconditional synchronization of startR and read we had in the former specification is recovered: $\{E^2_i\}_{i\in\{0,1,2,3\}}$ *as above but where* $E^2_0 = \{3.do(startR, s) \triangleright 1.do(read, s)\}$.

3. Finally, $\mathcal{E}' = \{E'_i\}_{i\in\{0,1,2,3\}}$ *such that* $E'_1 = E'_2 = \emptyset$,

$$E'_0 = \{3.\neg holds(r?, [s]) \to (3.do(startR, s) \rhd 1.do(read, s))\}$$

$$E'_3 = \{\neg holds(r?, [\vec{S_0}])\}$$

and $\mathcal{E}'' = \{E''_i\}_{i \in \{0,1,2,3\}}$ *such that* $E''_1 = E''_2 = \emptyset$,

$$E''_0 = \{3.do(startR, s) \rhd 1.do(read, s)\} \text{ and } E''_3 = \{\neg holds(r?, [\vec{S_0}])\}$$

are explanations for the problem

$$\forall s \; \vec{actual}(s) \to (1.holds(r?, [s_{\downarrow 1}]) \leftrightarrow 3.holds(r?, [s_{\downarrow 3}])).$$

The explanation $\mathcal{E}''$ in Example 14 can only be obtained when working at community level since it includes an interaction formula.

5. Local and global knowledge

We follow [24, 17] when formalizing knowledge in the situation calculus. At local level, knowledge is modeled by an accessibility relation on possible worlds, which is used to characterize information about what an agent knows. The knowledge of an agent can be enriched by doing knowledge producing actions. We propose an extension to multi-agent systems following, as before, a modular specification strategy.

5.1. Local knowledge

In this subsection we follow some of the ideas presented in [24, 17]. It is assumed a binary predicate K on situations, where $K(s', s)$, read s' is accessible from s, means that in situation s the agent thinks the world could be in situation s'. Something is known in s if is it true in every situation s' accessible from s. Then, the following abbreviation is introduced, where q is a state constraint:

$$\mathbf{Knows}(q, s) =_{\text{abv}} \forall s' \; (K(s', s) \to q_{[s']}).$$

Hence, q is known in s if and only if q holds in all situations s' accessible from s. In addition to the regular actions, considered before, it is also necessary to consider *knowledge producing actions*, i.e., actions whose effect is to make known the truth value of some formula. The occurrence of an action, knowledge or non-knowledge producing, affects the predicate K. The corresponding effects are defined by valuation axioms. If a is a non-knowledge producing action then the valuation axiom is

$$\forall s, s' \; (occurs(a(\vec{x}), s) \wedge K(s', s) \wedge \alpha_{[s]}{}^{f_i}_{p_i}) \to$$

$$(K(do(a(\vec{x}), s'), do(a(\vec{x}), s)) \wedge \alpha_{[do(a(\vec{x}),s)]}).$$

The idea is that the only knowledge produced in moving from s to $do(a(\vec{x}), s)$ is the knowledge that action a has occurred. Consider a knowledge producing action a_f, associated to a fluent f, in the sense that a_f is used by the agent to determine whether f is true. The valuation schema is as follows:

$$\forall s, s' \; occurs(a_f(\vec{x}), s) \wedge K(s', s) \wedge \alpha_{[s]} \rightarrow (K(do(a_f(\vec{x}), s'), do(a_f(\vec{x}), s)) \wedge \alpha_{[do(a_f(\vec{x}),s)]} \wedge (holds(f, [s]) \leftrightarrow holds(f, [s']))).$$

The idea is that in moving from s to $do(a_f(\vec{x}), s)$, the agent not only knows that $do(a_f(\vec{x}), s)$ has occurred but also the truth value of f. Furthermore, as usual in the literature, we assume that knowledge producing actions do not change the value of the attributes. In particular, f holds at $[do(a_f(\vec{x}), s)]$ iff f holds at $[s]$. However, this is not imposed by any restriction of our approach. Note that the agent specification definition needs to be relaxed in order to comply with the previous valuation schemata. The agent signature should also be adapted to cope with knowledge producing actions and non-knowledge producing actions. Furthermore, instead of successor state axioms for K, we propose valuation schemata, already discussed in Section 3.

EXAMPLE 15. *Consider a variant of the example in [17, 24]. Herein, we consider a distributed version of the safe problem with three agents: a* burglar *(b), a* safe *(s) and a* piece of paper *(p). The burglar has a piece of paper with a safe combination and a knowledge producing action read for reading the paper. The safe can only be opened if the correct combination is dialed. For simplicity, we assume that the combination is a natural number. The specification of the three agents is as follows:*

– $ASig^b = \langle\{read, opensf\}, \{comb, rich\}, \tau^b\rangle$ *where* $\tau^b(read) = \tau^b(opensf) = \tau^b(comb) = nat$, $\tau^b(rich) = \epsilon$;

  $ASpec^b = \langle ASig^b, AxF^b, AxS^b\rangle$ *where*

  - $AxF^b = \{poss(read(x), [s]) \rightarrow holds(comb(x), [s]),$
    $poss(opensf(x), [s]) \rightarrow \boldsymbol{Knows}(comb(x), s)\}$;
  - $AxS^b = \{\forall s, s' \; (occurs(read(x), s) \wedge K(s', s) \wedge \alpha_{[s]}) \rightarrow$
    $(K(do(read(x), s'), do(read(x), s)) \wedge \alpha_{[do(read(x),s)]} \wedge$
    $(holds(comb(x), [s']) \leftrightarrow holds(comb(x), [s]))),$
    $\forall s, s' \; (occurs(opensf(x), s) \wedge K(s', s) \wedge \alpha_{[s]}{}_{t}^{rich}) \rightarrow$
    $(K(do(opensf(x), s'), do(opensf(x), s)) \wedge \alpha_{[do(opensf(x),s)]})\}$;

– $ASig^s = \langle\{dcomb\}, \{comb, open\}, \tau^s\rangle$ *where* $\tau^s(dcomb) = \tau^s(comb) = nat$, $\tau^s(open) = \epsilon$;

$ASpec^s = \langle ASig^s, AxF^s, AxS^s\rangle$ *where*

- $AxF^s = \{\neg holds(open, [S_0]), holds(comb(C), [S_0]),$
  $poss(dcomb(x), [s]) \rightarrow holds(comb(x), [s])\}$;
- $AxS^s = \{\forall s, s' \; (occurs(dcomb(x), s) \wedge K(s', s) \wedge \alpha_{[s]}{}^{open}_{t}) \rightarrow$
  $(K(do(dcomb(x), s'), K(do(dcomb(x), s)) \wedge \alpha_{[do(dcomb(x),s)]})\}$;

– $ASig^p = \langle\{read\}, \{info\}, \tau^p\rangle$ *where* $\tau^p(read) = \tau^s(info) = nat$;

$ASpec^p = \langle ASig^p, AxF^p, AxS^p\rangle$ *where*

- $AxF^p = \{poss(read(x), [s]) \rightarrow holds(info(x), [s])\}$;
- $AxS^p = \{\forall s, s' \; (occurs(read(x), s) \wedge \alpha_{[s]} \wedge K(s', s)) \rightarrow$
  $(K(do(read(x)), s'), K(do(read(x), s)) \wedge \alpha_{[do(read(x),s)]})\}$.

Consider the burglar: to be able to open the safe, he must **know** *a combination. From the local point of view, this is the only information we can express. Later, when we embed the agent in the community, this combination must match the combination of the safe. As in [24],* $poss(opensf(s), [S_0])$ *does not follow from the burglar specification. However, from the knowledge producing action read:*

$$\begin{aligned}&\forall s \; (occurs(read(x), S_0) \wedge K(s, S_0)) \rightarrow \\ &\qquad (K(do(read(x), s), do(read(x), S_0)) \wedge \\ &\qquad (holds(comb(x), [s]) \leftrightarrow holds(comb(x), [S_0]))).\end{aligned} \tag{14}$$

The fluent comb is not altered by any action. Hence, we can infer the following result that will be useful when studying community properties:

$$\boldsymbol{Knows}(comb(x), do(read(x), S_0)). \tag{15}$$

5.2. Global knowledge

We extend the previous concepts to $CSpec = \langle CSig, AxF, AxS\rangle$ where $CSig = \langle ID, Sig\rangle$. As before, we want this extension to follow the modular specification strategy. We choose to represent the knowledge of the community through the local knowledge of each agent. To this end, we introduce an accessibility relation $\vec{K}$ on global situations

$$\vec{K}(s', s) =_{\text{abv}} \bigwedge_{i \in ID} i.K(s'_{\downarrow i}, s_{\downarrow i})$$

with the same intended meaning and the predicate $\vec{\mathbf{Knows}}$ as follows:

$$\vec{\mathbf{Knows}}(q, s) =_{\text{abv}} \forall s' \; \vec{K}(s', s) \to q_{[s']}.$$

The representation for global knowledge was not casual: we want to ensure that the knowledge of each agent carries over to the community.

PROPOSITION 13. *$i.\boldsymbol{Knows}(q, s_{\downarrow i}) \to \vec{\boldsymbol{Knows}}(i.q, s)$ holds for $i \in ID$ and $q \in SC_{Sig(i)}$.*

Proof. Assume that $i.\mathbf{Knows}(q, s_{\downarrow i})$ holds and let $\vec{K}(s', s)$. Then, by definition of $\vec{K}$, $i.K(s'_{\downarrow i}, s_{\downarrow i})$ holds. From the hypothesis, it follows that $i.q_{[s'_{\downarrow i}]}$ also holds, which implies (recall the comments after the definition of global state constraint) that $q_{[s']}$ holds. Hence, $\vec{\mathbf{Knows}}(i.q, s)$ holds.

In general, the converse does not hold because, even if $i.K(s'_{\downarrow i}, s_{\downarrow i})$ holds, it does not necessarily imply $\vec{K}(s', s)$ as it might be the case that, for some other agent $j \in ID$, $j.K(s'_{\downarrow j}, s_{\downarrow j})$ does not hold. This should not be unexpected. It is reasonable to assume that, when we put an agent in a community, information that was not known at local level, emerges by the interaction between agents. It is not very difficult to see that the structure of the state constraints is preserved by $\vec{\mathbf{Knows}}$ in the sense that if q is, for instance, of the form $i.p \to j.p'$ then

$$(i.\mathbf{Knows}(p, s_{\downarrow i}) \to j.\mathbf{Knows}(p', s_{\downarrow j})) \to \vec{\mathbf{Knows}}(i.p \to j.p', s).$$

The definition of community signature should also be relaxed. In particular, we want AxF_0 to possibly include some knowledge formulae involving more than one agent. This is the case of the next example.

EXAMPLE 16. *Recall Example 15. Besides the expected interactions, at the community level there is the knowledge that the combination in the paper corresponds to the safe combination. The community is: (i) $CSig = \langle \{1, 2, 3\}, Sig \rangle$ with $Sig(1) = ASig^b$, $Sig(2) = ASig^s$, and $Sig(3) = ASig^s$; (ii) $CSpec = \langle CSig, AxF, AxS \rangle$ with $AxF_1 = AxF^b$, $AxF_2 = AxF^s$, $AxF_3 = AxF^p$, $AxS_1 = AxS^b$, $AxS_2 = AxS^s$, $AxS_3 = AxS^p$ and*

$$\begin{aligned} AxF_0 = \{ & 1.do(read(x), s) \rhd 3.do(read(x), s), \\ & 1.do(opensf(x), s) \rhd 3.do(dcomb(x), s), \\ & \vec{\boldsymbol{Knows}}(3.info(x) \leftrightarrow 2.comb(x), \vec{S}_0)\} \end{aligned}$$

We want to conclude that the burglar will be able to open the safe. We may only conclude this at the global level. First we relate the information in the paper with the safe combination. We start by observing that

$\forall u$ holds$(comb(C), u)$, i.e., the safe combination does not change. This is a safety property that can be proved with the techniques of Section 3. Assume that $\vec{K}(s, \vec{S}_0)$ holds. Then, from the interaction axiom, we conclude $3.holds(info(x), [s_{\downarrow 3}]) \leftrightarrow 2.holds(comb(x), [s_{\downarrow 2}])$, which, together with the above safety property, implies $3.holds(info(C), [s_{\downarrow 3}])$. Hence,

$$\vec{\boldsymbol{Knows}}(3.info(C), \vec{S}_0) \tag{16}$$

Then, we relate the information on the paper with the information the burglar has after reading it. Consider the global action $a_1(x) = \langle read(x), nil, read(x) \rangle$. From (14), in particular:

$$\forall s \; (\vec{occurs}(a_1(x), \vec{S}_0) \wedge \vec{K}(s, S_0)) \rightarrow \tag{17}$$

$$(1.holds(comb(x), [s]) \leftrightarrow 1.holds(comb(x), [\vec{S}_0]))$$

Assume that $\vec{K}(s, \vec{do}(a_1(x), \vec{S}_0)$. Then, from (16) and a safety property for info stating that the value of the fluent does not change, it follows $3.holds(info(C), [s_{\downarrow 3}])$. This, together with the enabling of read implies that $x = C$, which together with the enabling of read implies $1.holds(comb(C), [s])$. Hence, using (17)

$$\vec{\boldsymbol{Knows}}(1.comb(C), \vec{do}(a_1(C), \vec{S}_0)) \tag{18}$$

Thus, we know that, after reading the paper, the burglar has the safe combination. The final step is to open the safe. We consider $a_2(x) = \langle opensf(x), dcomb(x), nil \rangle$. Using (18), the enabling on opensf holds only for $x = C$. But this coincides with the enabling of dcomb, together with the fact that the value of comb is not changed. Hence, $a_2(C)$ may occur since it is enabled and it is immediate to conclude that in the resulting state the safe is open, i.e., $2.holds(open, [\vec{do}(a_2(C), \vec{do}(a_1(C), S_0))_{\downarrow 2}])$.

Techniques of Section 3 for reasoning about safety still apply in the presence of knowledge. Further investigation is needed for extending the abduction techniques of Section 4 when in the presence of knowledge.

6. Conclusions

We addressed the issue of the specification and certification of multi-agent systems in the context of the situation (and state) calculus [21]. Our goal was to establish a modular framework: a system specification is a collection of local specifications (one for each agent), plus a specification of the interactions among the agents (synchronous action calling). We started with a simplified version of agent, where we only considered as specification axioms formulae describing initial conditions, the effects of actions on attributes and preconditions (enabling conditions) on action occurrence. We first presented some techniques for certification at local level. Then, we developed similar techniques for global level, capitalizing on the work previously developed, since our modular approach allows a uniform treatment of both levels. Similar ideas were presented in [25, 26] in the context of temporal logic. Certification approaches do not usually provide help on how a specification can be enriched when some intended property does not hold. Herein, we presented abductive reasoning techniques to tackle this problem (considering Boolean fluents). These techniques provide suitable new axioms for enriching a specification ensuring that it fulfills the envisaged requirement. Again, we started by working at local level, where the specification was enriched with suitable new initial conditions and enabling conditions. Then, we extended these techniques to deal with global properties. In this case, besides new initial and enabling conditions, we also can provide new interactions between agents. We focused on safety properties, but other properties can also be considered [8]. To our knowledge, this abductive approach to certification is a novel one in this context of multi-agent system specification. Finally, we extended the notion of agent in order to cope with knowledge. At local level, our approach was based in [24]. Then, we extend these knowledge features to the community, providing a uniform treatment of knowledge at both levels. Other approaches to the specification of multi-agent systems in the context of situation calculus have been proposed [13, 9]. However, the framework presented therein is not a modular one. In [9], specifications are plain, i.e., neither fluents nor actions are encapsulated. The language imposes no restrictions on which actions an agent can perform. Moreover, an agent is aware of the actions that are being performed by other agents (although these actions are hidden under the concept of exogenous actions). In [13] interaction between agents is dealt with by introducing *pseudo-fluents* in the language (for representing communication channels). In our approach, no new symbols (fluents or actions) are needed to represent interaction among agents.

As future work we intend to extend our certification and abductive techniques to deal with properties other than safety, namely, liveness properties, following the work already developed in [20, 8]. In addition, we intend to develop abductive techniques for multi-agent systems specifications including also knowledge.

Acknowledgements

This work was partially supported by FCT and FEDER, namely, via the Project FibLog POCTI/MAT/372 39/2001 of CLC. The authors also like to thank the anonymous referees for their valuable comments.

References

1. Ben-Ari, M.: 1990, *Principles of Concurrent and Distributed Programming.* Prentice Hall International.
2. Ciancarini, P. and M. Wooldridge: 2001, 'Agent-Oriented Softwrae Engineering'. In: *Lecture Notes in AI*, Vol. 1957. Springer-Verlag.
3. Dix, J., H. Munoz-Avila, and D. Nau: 2003, 'IMPACTing SHOP: Putting an AI Planner into a Multi-Agent Environment'. *AMAI.* to appear.
4. Fisher, M. and C. Gidini: 2002, 'Agents with Bounded Temporal Resources'. In: *Foundations and Applications of Multi-Agent Systems*, Vol. 2403 of *Lecture Notes in Computer Scince.* Springer-Verlag, pp. 169–184.
5. Fitting, M.: 1996, *First-Order Logic and Automated Theorem Proving.* Springer, New York.
6. Gelfond, M., V. Lifschitz, and A. Rabinov: 1991, 'What are the limitations of the situation calculus?'. In: R. Boyer (ed.): *Automated Reasoning: Essays in Honor of Woody Bledsoe.* Kluwer Academic, pp. 167–179.
7. Gouveia, P.: 1998, 'Abductive Reasoning over Temporal Specifications of Objects'. Ph.D. thesis, IST. In Portuguese.
8. Gouveia, P. and C. Sernadas: 2001, 'Abduction reasoning over temporal specifications of objects'. In: M. Zakharyaschev, K. Segerberg, M. de Rijke, and H. Wansing (eds.): *Advances in modal logic - volume 2.* CSLI, pp. 275–300.
9. Lespérance, Y., H. Levesque, and R. Reiter: 1999, 'A Situation Calculus Approach to Modeling and Programming Agents'. In: A. Rao and M. Wooldridge (eds.): *Foundations and Theories of Rational Agency.* Kluwer.
10. Lin, F. and R. Reiter: 1994, 'State constraints revisited'. *Journal of Logic and Computation* **4**(5), 655–678.
11. Manna, Z. and A. Pnueli: 1991, 'Completing the temporal picture'. *Theoretical Computer Science* **93**, 97–130.
12. Manna, Z. and A. Pnueli: 1995, *The Temporal Logic of Reactive and Concurrent Systems: Safety.* Springer Verlag.
13. Marcu, M., Y. Lespérance, H. Levesque, F. Lin, R. Reiter, and R. Scherl: 1995, 'Distributed Software Agents and Communication in the Situation Calculus'. In: *In Proc. Intelligent Computer Communication (ICC'95).* Romania.
14. Mateus, P., A. Pacheco, J. Pinto, A. Sernadas, and C. Sernadas: 2001, 'Probabilistic Situation Calculus'. *Annals of Math. and AI* **32**(1/4), 393–431.
15. Mayer, M. and F. Pirri: 1993, 'First order abduction via tableau and sequent calculi'. In: *Bulletin of the IGPL 1 (1).* pp. 99–117.
16. McCarthy, J. and P. Hayes: 1969, 'Some philosophical problems from the standpoint of Artificial Intelligence'. In: B. Meltzer and D. Michie (eds.): *Machine Intelligence 4.* Edinburgh University Press, pp. 463–502.

17. Moore, R.: 1985, 'A formal theory of knowledge and action'. In: J. Hoobs and R. Moore (eds.): *Formal Theories of the Commonsense Wrold*. Ablex Publishing, pp. 319–358.
18. Peirce, C.: 1955, 'Abduction and induction'. In: Buchler (ed.): *Philosophical Writings of Peirce*. Dover, pp. 150–156.
19. Pinto, J. and R. Reiter: 1995, 'Reasoning about time in the situation calculus'. *AMAI: Papers in Honour of Jack Minker* **14**(2-4), 251–268.
20. Ramos, J.: 2000, 'The Situation and State Calculus: Specification and Verification'. Ph.D. thesis, IST, Universidade Técnica de Lisboa.
21. Ramos, J. and A. Sernadas: 1999, 'The situation and state calculus versus branching temporal logic'. In: J. Fiadeiro (ed.): *Recent Developments in Algebraic Development Techniques, Selected Papers*, Vol. 1589 of *LNCS*. Springer-Verlag, pp. 293–309.
22. Reiter, R.: 1991, 'The frame problem in the situation calculus: a simple solution (sometimes) and a completeness result for goal regression'. In: V. Lifschitz (ed.): *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*. San Diego, CA: Academic Press, pp. 359–380.
23. Reiter, R.: 1993, 'Proving properties of states in the situation calculus'. *Articial Intelligence* **64**, 337–351.
24. Scherl, R. and H. Levesque: 1993, 'The Frame Problem and Knowledge-Producing Actions'. In: *Proc. of 11th National Conf. on AI*. pp. 689–697.
25. Sernadas, A., C. Sernadas, and J. Costa: 1995, 'Object specification logic'. *Journal of Logic and Computation* **1**, 7–25.
26. Sernadas, A., C. Sernadas, and J. Ramos: 1996, 'A temporal logic approach to object certification'. *Data and Knowledge Engineering* **19**, 267–294.
27. Thielscher, M.: 1999, 'From situation calculus to fluent calculus: state update axioms as a solution to the inferential frame problem'. *Artificial Intelligence* **111**(1-2), 277–299.
28. van der Hoek, W. and W. Wooldrige: 2003, 'Towards a Logic of Rational Agency'. *Logic Journal of the IGPL* **11**(2), 133–157.