Role Monitoring in Open Agent Societies

Federico Chesani, Paola Mello, Marco Montali, and Paolo Torroni

DEIS, University of Bologna.
V.le Risorgimento 2
40136 Bologna, Italy
{federico.chesani,paola.mello,marco.montali,paolo.torroni}@unibo.it

Abstract. We address run-time monitoring of membership, roles and role dynamics in open agent societies. To this end, we build on Dignum's formalization of agent organizations, on the SOCS computational logic agent framework and on the Event Calculus for representing and reasoning about time.

1 Introduction

The concept of *role* is recognized in the Multi Agent Systems (MAS) community as a fundamental element of agent organisations and societies. A role is characterized by (*i*) the way an agent begins/terminate to enact that role; (*ii*) the capabilities, rights and duties associated with the role; and (*iii*) the organizational structure that relate each role with the other ones (also referred to as the *power* structure among the roles).

The way an agent begins and terminates to enact a certain role is strictly related to the openness degree of an agent society. Open societies (i.e., following Davidsson [1], societies in which agents can freely join and leave such organisations) are characterized by a high variability of the agents taking part to it, while closed societies are more uniform in this sense. Either way, it is often the case that the same role is enacted by several different agents, e.g., as a consequence of interactions among the participants. Role playing is in general a very dynamic concept, which calls for methods for tracking role changes, and to know at each instant which are the roles an agent is playing.

Roles are characterized also by the capabilities, rights, and duties they bring along. For example, in the Gaia agent-oriented methodology [2] a role is strictly linked to a set of *rights* (what an agent embodying the role can do) and *responsibilities* (what an agent embodying the role should do). Responsibilities then can be of two types: *liveness* (i.e., ensuring that something good will happen) and *safety* (i.e., something bad will not happen). In [3] the notion of roles is instead associated with the concepts of *capabilities* [1] (actions that an agent enacting the role can perform) and *expected behavior* (i.e., which is the expected reaction to

[1] Note that the *capabilities* associated with a role here are intended as the actions that an agents (enacting that role) is allowed to perform. However, the term "capabilities" is also used with reference to an agent to indicate the "skills" of the agent itself.

P. Jędrzejowicz et al. (Eds.): KES-AMSTA 2010, Part I, LNAI 6070, pp. 112–121, 2010.

incoming events). While capabilities are used to characterize the pro-activeness of an agent (since they concern what an agent may do), expected behaviour is more related to the agent reactiveness. In [4,5] an "institutional" approach is assumed: it is the institution itself that is responsible of controlling agents, and to ensure the expected behaviour is achieved coherently with the specification of the corresponding roles (possibly prohibiting certain interactions at the infrastrcture level). Again, at the society level is of the utmost importance to know, at each instant, which agent is enacting a certain role.

The third aspect that characterizes the notion of role is the "power" structure, intended as the way agents coordinate each other and delegate the tasks. This dimension, as pointed out in [6], is strictly related to the coordination and the organizational aspects of the agent society. For example, in a hiearchical society an agent enacting the role of "boss" can delegate tasks to an agent enacting the role of "phdStudent". The delegation act, together with the (vested) power relation between the roles of boss and Ph.D. student, bring about an obligation (the duty) on the side of the Ph.D. student, to achieve the assigned tasks. Network and market-like organizations instead rely on some coordination mechanism (usually message exchange) in order to establish the tasks delegation among the agents. In such cases, the obligations are a consequence of coordination actions among the agents. A possibility is that an agent, in order to perform a certain task, acquires (starts to enact) a certain role, temporarily and in a dynamic way.

In this paper we build on previous work on agents societies, and in particular on the $\mathcal{S}$CIFF framework [7]. Such a framework already supports natively the concepts of obligations and prohibitions at the society level, but it lacks the notion of dynamic role. To overcome such a limitation, we exploit an extension of the Event Calculus (EC) [8] written in $\mathcal{S}$CIFF, called Reactive Event Calculus ($\mathcal{REC}$, [9]).

We show how these extensions allow us to declaratively define, represent and reason upon the dynamic relations between agents and roles, by means of *fluents*, i.e., fluents are properties that can change their value in time as events happen. Domain axioms, which are a part of a MAS specification, state how fluents change their value depending on other fluents/conditions and on event occurrences. The $\mathcal{EC}$ offers a very simple, compact yet powerful notation for specifying the relation between events and fluents, and it has many other interesting features such as symmetry of past and future, generality with respect to time orderings, executability and direct mapping with computational logic frameworks, immunity from the frame problem, and explicit treatment of time and events.

$\mathcal{EC}$ implementations have been mainly used for a-posteriori reasoning upon an observed event narrative, or for planning courses of events that lead to reach a desired state (a goal). $\mathcal{REC}$ can also successfully support run-time monitoring of fluents. $\mathcal{S}$CIFF and $\mathcal{REC}$ together enable continuous run-time monitoring of an agent society's evolution, both in terms of roles, and in terms of obligations/prohibitions.

The paper is structured as follows. In Section 2 we give the necessary background about the $\mathcal{S}$CIFF language and proof-procedure, and the $\mathcal{REC}$. We assume that the reader is familiar with the EC. In Section 3 we show how to define roles, relations between agents and roles, power structure and obligations. In Section 4 we adopt an example inspired by the discussion in [6], to illustrate our approach and its monitoring features. Section 5 concludes.

2 Preliminaries

$\mathcal{S}$CIFF [7] is an extension of Fung an Kowalski's IFF proof-procedure for abductive logic programming [10,11], acommodating the representations of social interactions among MAS. It uses two primitive notions: events (mapped as **H** atoms) and expectations (mapped as **E**/**EN** atoms). $\mathbf{H}(Ev, T)$ means that an event Ev has occurred at time T, and it is a ground atom. Instead $\mathbf{E}(Ev, T)$ and $\mathbf{EN}(Ev, T)$ can contain variables with domains and CLP constraints, and they denote in the first case that an event unifying with Ev is expected to occur at some time in the range of T (T being existentially quantified), and in the second case that all events unifying with Ev are expected to not occur, at all times in the range of T (i.e., T is considered as universally quantified over its CLP range). **E** atoms represent obligations, while **EN** represent prohibitions. $\mathcal{S}$CIFF accommodates existential and universal variable quantification and quantifier restrictions, CLP constraints, dynamic update of event narrative and it has a built-in runtime protocol verification procedure.

A $\mathcal{S}$CIFF specification is composed of a knowledge base KB, a set of integrity constraints (ICs) $\mathcal{IC}$, and a goal $\mathcal{G}$. KB consists of backward rules $head \leftarrow body$, whereas the ICs in $\mathcal{IC}$ are forward implications $body \rightarrow head$, which must be satisfied at all times (see for example Eq. 3). ICs are interpreted in a reactive manner. To put it simply, when the body of an IC becomes true (i.e., the involved events occur), then the rule fires, and the expectations in the head are generated by abduction. For example, $\mathbf{H}(a, T) \rightarrow \mathbf{EN}(b, T')$ defines a relation between events a and b, saying that if a occurs at time T, b should not occur at any time; $\mathbf{H}(a, T) \rightarrow \mathbf{E}(b, T') \wedge T' \leq T + 300$ says that if a occurs, then an event b should occur no later than 300 time units after a.

To exhibit a correct behavior, given a goal $\mathcal{G}$ and a triplet $\langle KB, \mathcal{A}, ICs \rangle$, a set of abduced expectations must be *fulfilled* by corresponding events. The $\mathcal{S}$CIFF semantics [7] is given for a given specifications and narrative, denoted by **HAP** (a set of **H** atoms), and it intuitively states that KB, together with the abduced literals, must entail $\mathcal{G} \wedge ICs$, **E** expectations must have a corresponding matching happened event, and **EN** expectations must not have a corresponding matching event. Moreover, **E** and **EN** atoms must be consistent and not contradictory (this is formalized in [7] via the **E**-consistency notion).

The Reactive Event Calculus ($\mathcal{REC}$) is an implementation of $\mathcal{EC}$ on top of the $\mathcal{S}$CIFF Framework. In particular, the $\mathcal{S}$CIFF axiomatization of $\mathcal{REC}$ draws inspiration from Chittaro and Montanari's *Cached Event Calculus* $\mathcal{CEC}$ [12] and on their idea of maximum validity intervals (MVIs). Events and fluents are terms

and times are integer (CLP) variables, 0 being the "initial" time. The main distinctive feature of $\mathcal{REC}$ is its reactivity: fluents are initiated and terminated by dynamically occurring events. Thus its name, "reactive event calculus" ($\mathcal{REC}$). Thanks to its reactivity $\mathcal{REC}$ overcomes the limit of many previous implementations of $\mathcal{EC}$ and it can be efficiently used for monitoring purposes.

$\mathcal{REC}$ uses the abduction mechanism to generate MVIs and define their persistence. It has a fully declarative axiomatization [9]: no operational specifications are needed. It uses two special internal events (denoted by the reserved *clip*/*declip* words, differently from generic external events, that are "incapsulated" into the reserved term *event*) to model that a fluent is terminated/initiated, respectively. $\mathcal{REC}$ models can be specified using usual $\mathcal{EC}$ notions like *initiates*, *terminates*, *initially_holds*, and *holds_at*.

3 Representing Dynamic Roles, Obligations and Prohibitions

We exploit $\mathcal{S}$CIFF and $\mathcal{REC}$ to represent and reason upon roles, norms, obligations and prohibitions. In particular, we will resort to the $\mathcal{S}$CIFF notions of positive/negative expectations for representing obligations/prohibitions respectively (as in [7,13]). In order to capture dynamic aspects like agents emobodying roles instead we will resort to the $\mathcal{REC}$ facilities.

We represent a role by means of a ground term[2]. For example, the following formulas specified in the knowledge base:

$$\begin{array}{l} role(boss). \\ role(phdStudent). \end{array} \tag{1}$$

assert that there exist two different roles, named boss and phdStudent.

Many approaches in the literature associate the notion of roles with many aspects, ranging from capabilities, to duties and powers. Through the knowledge base, it is possible to specify all/part of the features that characterize the specific role. We do not suggest here which features should be/should not be represented explicitly in the knowledge base. Our solution allows enough flexibility to let each user specify the notion of role at the desired level of detail. In the reminder of the paper we will represent only the power structure, i.e. the notions of power among roles from which are generated the obligations for agents embodying that roles. Note that we are using the terminology "power structure" as in [6]. For example, a boss has the power to assing tasks to his phdStudent.

$$power(boss, phdStudent, assignTask(X)).$$

Note that the *assignTask* term has a parameter, specified in the formula above with a unbounded variable. In our framework this means that a boss can assign

[2] Here and in the following, we will adopt a Prolog-like notation, where terms starting by a capital letter indicate unbound variables (not yet assigned to any value), while terms starting by a lower case letter indicate objects (concepts) of our domain.

to a student *any* task, and that from that event (following [6]) an obligation arise for the student.

Strictly linked to the notion of role there is the notion of *role enacting agent* (*rea*), i.e. the relation that dynamically links agents and roles. Such relation has many characteristics: an agent can play many different roles, and the same role can be played by many different agents. An agent can be appointed to play a role *statically*, i.e. independently of any particular interaction or execution course. An agent can also assume a role *dynamically*, as a consequence of certain actions he/she performed. Moreover, an agent could begin/terminate to enact a role (as of a certain point), as a consequence not only of its own actions, but also as a consequence of other agents' actions. Such information is represented by means of fluents, which can change their truth value along the temporal dimension (depending on the event narrative). Moreover, thanks to the *initially* axioms in the $\mathcal{EC}$, fluents can be used to model organizations in which structure and roles are statically defined, and do not change in time. For example, to represent the fact that an agent named john is statically playing the role of boss, it is sufficient to assert in the knowledge base the fact:

$$initially\_holds(rea(john, boss)).$$

john retains his role as long as no termination axioms are defined in the knowledge base. Suppose we would like to model the fact that, upon retirement, an agent (e.g., john) will cease to embody the boss role. It is sufficient to extend the knowledge base as follow:

$$terminates(retirement(X, T), rea(X, boss), T).$$

The process of entering/leaving the societies is another dynamic element in the society. In semi-open societies [1] agents can join the MAS provided they accomplish some tasks. For example, an agent david will start enacting the role phdStudent after he is hired by a boss. In this case, an *initiates* axiom can be used:

$$\begin{aligned} initiates(hires(X, Y), rea(Y, phdStudent), T) \leftarrow \\ holds\_at(rea(X, boss), T). \end{aligned} \tag{2}$$

The above equation states that the event of "X hiring Y at time T" makes Y enacting the role of phdStudent, provided that X is a boss at time T.

Obligations and prohibitions are already supported by the $\mathcal{S}$CIFF framework. By adding fluents to an IC's body, it is possible to link dynamic roles and obligations that arise from role specifications. For example, suppose we want to state that once a boss has assigned a task to a Ph.D. student, then the student has an obligation to fulfil that task:

$$\begin{aligned} &\mathbf{H}(X, Y, assignTask(reviewPaper(id51)), T) \wedge \\ &holds\_at(rea(X, X\_role), T) \wedge holds\_at(rea(Y, Y\_role), T) \wedge \\ &right(X\_role, Y\_role, assignTask(reviewPaper(IDPaper)) \rightarrow \\ &\quad \mathbf{E}(Y, X, provideReview(IDPaper, Comments), T_1). \end{aligned} \tag{3}$$

Eq. 3 states that, if the *assignTask* event happens, and if roles of agent X and Y are X_role and Y_role respectively, and if there is a power structure that establishes the right of X_role agents to asign tasks to Y_role agents, then an obligation for agent Y follows.

4 Monitoring Dynamic Roles, Obligations, and Prohibitions

One of the most advanced features of the $\mathcal{REC}$-enhanced $\mathcal{S}$CIFF framework is its support to runtime monitoring. Both the frameworks share the same underlying proof-procedure, that allows to automatically check if expectations (positive and negative) are indeed fulfilled. In particular, a major feature of $\mathcal{REC}$ is the possibility of monitoring fluents values, showing at each instant which fluent holds. Let us introduce an example freely inspired by the one discussed in [6].

Example 1. Full professor john is a member of the program committee of a conference, and he has been requested to review the paper id51. He has also hired a new Ph.D. student, david, just a few days after he received the paper. john decides that his student could learn a lot by helping him reviewing the paper, so he assigns david the review. However, john does not fully trust david's judgement, so he asks marc, a postdoc, to review the paper anyway: as a reward, john and marc agree on a dinner in the best restaurant of the city.

This simple example contains many different aspects mixed together: there are static, defined a-priori roles, as well as dynamic roles; there are obligations that arise as a consequence of a power structure among roles; and there are obligations that arise as a consequence of a market-like mechanism. Example 1 can be easily represented in our approach in the following way. Eq. 4 specifies that john is embodying the role of boss, while marc is embodying the role of postdoc. Eq. 5 shows a possible definition of the power structure: an agent playing the boss role has the power to hire Ph.D. student, as well as to assign papers to them.

$$\begin{aligned} &initially\_holds(rea(john, boss)).\\ &initially\_holds(rea(marc, postdoc)). \end{aligned} \tag{4}$$

$$\begin{aligned} &power(boss, hirePhdStudent).\\ &power(boss, phdStudent, assignPaper(\_, \_, \_)). \end{aligned} \tag{5}$$

Eq. 6 specifies the conditions by which an agent Y assumes the role of phdStudent dynamically: there must be an agent X that hires Y, and moreover X should emobdy a role $XRole$ that has the power to hire a Ph.D. student. Note that among the conditions, the boss should have also got financing for the student (it is required that an event $gotmoney(\ldots)$ happened before hiring the student).

Eq. 7 instead is a $\mathcal{S}$CIFF Integrity Constraint, where the happened events (an agent X assigning to an agent Y a paper to be reviewed), the conditions (X and Y having a certain role), and the power structure among the roles enable some obligations (Y is obliged to provide a review to X).

$$\begin{aligned} &initiates\_at(hires(X,Y), rea(Y, phdStudent), T) \leftarrow \\ &\quad holds\_at(rea(X, XRole), T), \\ &\quad power(XRole, hirePhdStudent), \\ &\quad h(event(gotMoney(X)), T_0), T_0 < T. \end{aligned} \tag{6}$$

$$\begin{aligned} &\mathbf{H}(assignPaper(X,Y,IDPaper), T) \\ &\wedge holds\_at(rea(X, XRole), T), \\ &\wedge holds\_at(rea(Y, YRole), T), \\ &\wedge power(XRole, YRole, assignPaper(X,Y,IDPaper)) \rightarrow \\ &\quad \mathbf{E}(provideReview(Y,X,IDPaper), T1). \end{aligned} \tag{7}$$

Finally, Eq. 8a and Eq. 8b establish the conditions by which agent Y has an obligation towards X to fulfill a certain task: in that case, the obligation is a consequence of a market-like agreement where Y accepts the task and receives a reward. Eq. 8a establishes the conditions for the obligation, while Eq. 8b establishes the conditions that must be fulfilled in order for Y to receive the reward: upon completion of the task, X has an obligation towards Y to provide him the reward.

$$\begin{aligned} &\mathbf{H}(agree(Y,X,Task,Reward), T) \\ &\wedge \mathbf{H}(event(proposeTask(X,Y,Task,Reward)), T_0) \rightarrow \\ &\quad \mathbf{E}(completeTask(Y,X,Task), T_1), \\ &\quad \wedge T_0 < T \wedge T < T_1. \end{aligned} \tag{8a}$$

$$\begin{aligned} &\mathbf{H}(completeTask(Y,X,Task), T_2) \\ &\wedge \mathbf{H}(event(proposeTask(X,Y,Task,Reward)), T_0) \\ &\wedge \mathbf{H}(event(agree(Y,X,Task,Reward)), T_1) \\ &\wedge T_0 < T_1 \wedge T_1 < T_2 \rightarrow \\ &\quad \mathbf{E}(provideReward(Reward)), T_3) \wedge T_2 < T_3. \end{aligned} \tag{8b}$$

Figure 4 shows the output computed by the $\mathcal{REC}$ when observing the event narrative listed in Table 1, which summarizes Example 1. Fluents representing the *rea* relation are in light blue (top three bars): for example, david starts embodying the phdStudent role after he is hired by his boss. Obligations (positive expectations in the $\mathcal{S}$CIFF Framework) are instead in orange (bottom three bars). To ease the understanding of the output, obligations have been drawn similarly to fluents. The difference is that positive expectations must be fulfilled by a corresponding event. In Figure 4 an expectation about david is raised up at time 20 (when the boss assign the paper), and it is satisfied (hence it is lowered) at time 43 (when david accomplishes the task). Note also that the obligation for the full professor to reward the postdoc is raised when marc completes the task he has agreed upon (i.e., at time 45).

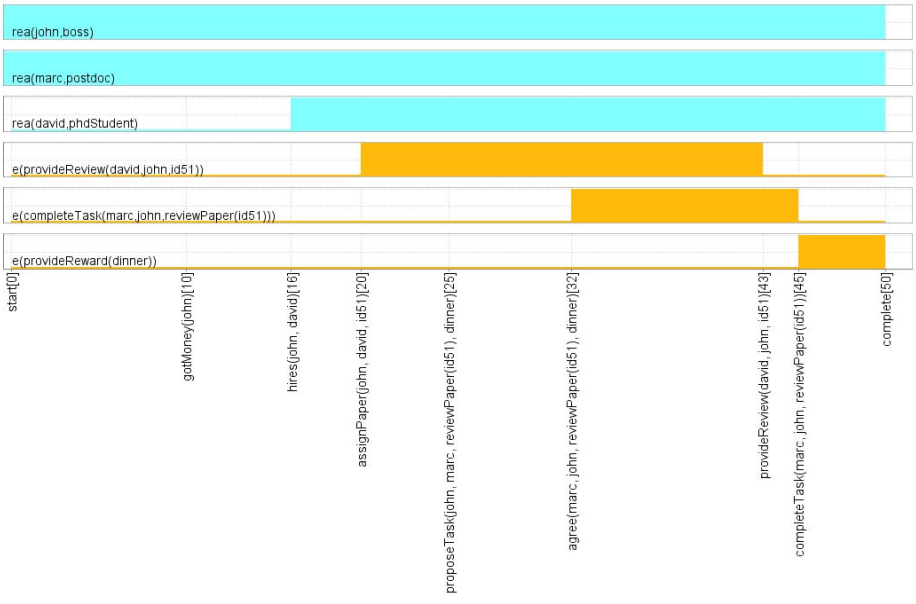

Fig. 1. Monitoring roles

Table 1. An example of an event narrative

Event	**Time**
gotMoney(john)	10.0
hires(john, david)	16.0
assignPaper(john, david, id51)	20.0
proposeTask(john,marc,reviewPaper(id51),dinner)	25.0
agree(marc, john, reviewPaper(id51), dinner)	32.0
provideReview(david, john, id51)	43.0
completeTask(marc, john, reviewPaper(id51))	45.0

5 Related Works and Conclusions

Roles are a fundamental concept for representing agent societies and organisations. In particular, dynamic roles are of the utmost importance to build open and flexible multi-agent systems. Dynamic roles however require the capability of monitoring them, in order to control and manage the agent society. In this paper we showed how to use $\mathcal{S}$CIFF and $\mathcal{REC}$ together to represent and reason upon roles, obligations and powers and their dynamics in time. We also showed how $\mathcal{S}$CIFF enables an effective run-time monitoring of agent roles and obligations. To the best of our knowledge, this is the first proposal in the literature that does it.

Note that $\mathcal{EC}$ has been used before for representing institutional and normative informations in MAS. For example, in [14] the authors use the $\mathcal{EC}$ to capture the normative state of affair of a (business) contract relationship among many partners, although they do not model roles. They exploit the $\mathcal{EC}$ to represent some of the standard normative concepts, and extend the classic $\mathcal{EC}$ by introducing the concept of *multivalued fluents* and obligation fulfillment/violation. In our approach such concepts are directly supported in the $\mathcal{S}$CIFF framework and, more important, all such concepts are equipped with a clear declarative semantics. Moreover, in our approach, obligations and prohibitions can be partially specified, i.e. the data structures representing them can contains unbound variables. In order to perform reasoning, in [14] an ad-hoc Java-based program has been implemented. $\mathcal{REC}$ instead relies on Prolog and on the $\mathcal{S}$CIFF proof-procedure, and hence it is equipped of formal proofs of soundness and completeness (w.r.t. its declarative semantics). Moreover, $\mathcal{REC}$ has been specifically developed to overcome classic limitations of other $\mathcal{EC}$ implementations, i.e. it is able to perform runtime reasoning, since its ability to cope with dynamically happening events.

Future work will be devoted to extending our framework's features. In particular, we have considered only obligations and prohibitions, and a very simple power structure. In this work we did not address permissions and other deontic operators usually adopted when reasoning about normative systems. On the implementation side, a Java-based interface is under development, which will enhance the interface bewteen the $\mathcal{S}$CIFF reasoner (written in SWI Prolog) and other commercial applications.

$\mathcal{S}$CIFF can be downloaded from its Web site: `http://lia.deis.unibo.it/sciff/`.

Acknowledgements. This work has been partially supported by the Italian MIUR PRIN 2007 project No. 20077WWCR8.

References

1. Davidsson, P.: Categories of artificial societies. In: Omicini, A., Petta, P., Tolksdorf, R. (eds.) ESAW 2001. LNCS (LNAI), vol. 2203, pp. 1–9. Springer, Heidelberg (2001)
2. Wooldridge, M., Jennings, N.R., Kinny, D.: The gaia methodology for agent-oriented analysis and design. Autonomous Agents and Multi-Agent Systems 3(3), 285–312 (2000)
3. Cabri, G., Leonardi, L., Zambonelli, F.: Modeling role-based interactions for agents. In: Debenham, J., Henderson-Sellers, B., Jennings, N., Odell, J. (eds.) OOPSLA 2002 Workshop, Seattle, USA (2002)
4. Esteva, M., de la Cruz, D., Sierra, C.: ISLANDER: an electronic institutions editor. In: Castelfranchi, C., Lewis Johnson, W. (eds.) AAMAS-2002, Part III, pp. 1045–1052. ACM Press, New York (2002)
5. Noriega, P., Sierra, C.: Electronic Institutions: Future Trends and Challenges. In: Klusch, M., Ossowski, S., Shehory, O. (eds.) CIA 2002. LNCS (LNAI), vol. 2446, pp. 91–99. Springer, Heidelberg (2002)

6. Dignum, V., Dignum, F.: Coordinating tasks in agent organizations. or: Can we ask you to read this paper? In: Noriega, P., Vázquez-Salceda, J., Boella, G., Boissier, O., Dignum, V., Fornara, N., Matson, E. (eds.) COIN 2006. LNCS (LNAI), vol. 4386, pp. 32–47. Springer, Heidelberg (2007)
7. Alberti, M., Chesani, F., Gavanelli, M., Lamma, E., Mello, P., Torroni, P.: Verifiable agent interaction in abductive logic programming: the SCIFF framework. ACM TOCL 9(4), 1–43 (2008)
8. Kowalski, R.A., Sergot, M.: A logic-based calculus of events. New Generation Computing 4(1), 67–95 (1986)
9. Chesani, F., Mello, P., Montali, M., Torroni, P.: Commitment tracking via the reactive event calculus. In: Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI). AAAI Press, Menlo Park (2009)
10. Fung, T.H., Kowalski, R.A.: The IFF proof procedure for abductive logic programming. Journal of Logic Programming 33(2), 151–165 (1997)
11. Kakas, A.C., Kowalski, R.A., Toni, F.: Abductive Logic Programming. Journal of Logic and Computation 2(6), 719–770 (1993)
12. Chittaro, L., Montanari, A.: Efficient temporal reasoning in the cached event calculus. Computational Intelligence 12(2), 359–382 (1996)
13. Alberti, M., Gavanelli, M., Lamma, E., Mello, P., Sartor, G., Torroni, P.: Mapping deontic operators to abductive expectations. Computational and Mathematical Organization Theory 12(2-3), 205–225 (2006)
14. Farrell, A.D.H., Sergot, M.J., Sallé, M., Bartolini, C.: Using the event calculus for tracking the normative state of contracts. Int. J. Cooperative Inf. Syst. 14(2-3), 99–129 (2005)