Acta Universitaria
ISSN: 0188-6266
actauniversitaria@ugto.mx
Universidad de Guanajuato
México

Acosta G., Juan C.; Marcial Romero, J. Raymundo; Ramos C., Marco A.; Hernández Servín, J. A.
Towards a Calendar Agent Society with Intelligent Agents in ASP-Updates.
Acta Universitaria, vol. 22, marzo, 2012, pp. 48-54
Universidad de Guanajuato
Guanajuato, México

Towards a Calendar Agent Society with Intelligent Agents in ASP-Updates.

Juan C. Acosta G.*, J. Raymundo Marcial Romero*, Marco A. Ramos C.* and J. A. Hernández Servín *

ABSTRACT

We present the analysis and some preliminary specifications to describe a multi-agent society to represent dynamic appointments under in Answer-Sets Programming, by means of knowledge-base updates. This is a case study to represent a group of intelligent BDI agents with a common goal of scheduling a meeting, and we use an example to show how to distribute and change their specifications under unforeseen circumstances, as well as a simple protocol to realise a consensual meeting. We claim that the proposed framework is appropriate to have the benefits from a strong foundation like Answer-Sets Programming, simplicity for its declarative logic programming, as well as practicality for existent implemented solvers, which can be used to implement a more-complete and useful system of agent societies.

RESUMEN

Se presenta el análisis y algunas especificaciones para describir una sociedad multi-agente que representa agendas dinámicas en Answer Sets Programming, por medio de actualización de bases de conocimiento. Se trata de un estudio de caso para representar un grupo de agentes BDI inteligentes con el fin común de programar una reunión, y se utiliza un ejemplo para mostrar la manera de distribuir y cambiar sus especificaciones bajo circunstancias imprevistas, así como un protocolo sencillo para realizar una reunión consensual. Sostenemos que el marco propuesto es apropiadopara tener los beneficios de una base sólida como Answer Sets Programming, de la simplicidad de su programación lógica declarativa, así como de la practicidad para resolvedores existentes implementados, que pueden ser utilizados para un sistema más completo y útil de sociedades de agentes.

Recibido: 10 de Enero de 2012
Aceptado: 14 de Febrero de 2012

INTRODUCTION AND MOTIVATION

Nowadays, we can find powerful calendar systems in computer industry to manage appointments and to plan tasks. They may classify appointments, work in groups, be updated and consulted anywhere in the world.
To the authors' knowledge, however, current commercial calendar systems[1] are still far from implementing mechanisms to gather a group of people with different preferences, for a common meeting or managing conflicting situations like scheduling more than one appointment at the same time!
One of the most *elegant* ways of some of such calendars is limited to saying that there is a conflict by means of a red line around the conflicting appointment, and the user should solve it as he or she can.
So, we need a way of producing a more elaborated suggestion from such conflicts, by taking into account the user's preferences and the other's intentions.

Palabras clave:
Calendario, programación de answer-sets, actualizaciones, programación lógica, sistemas multi-agente
Keywords:
Calendar; answer-sets programming; updates; logic programming; multi-agents system

In Artificial Intelligence, for instance, several proposals aim at the solution of the problem to manage coherent calendars with multiple users' agents and common meetings: [1, 2, 3]. They use sophisticated algorithms and a mixture of frameworks to overcome the limitation of current industrial approaches, with solid ordinary systems. However, if we are to build autonomous intelligent agents in general, we need to provide them both with an evolving knowledge base (KB) and guaranteed autonomy so as to change not only information, but even their very specifications and configuration.These features from the literature ought to allow them both to learn naturally from their environment and to sense it with a corresponding reaction.

* U. Autónoma del Estado de México, FI-UAEM . Toluca, México. E-mail: jguadarrama@gmail.com, jrmarcialr@uaemex.mx
[1] In our opinion, some of the most popular calendars are Google Calendar™; Yahoo Calendar™; Apple iCal™ (http://www.apple.com/macosx/features/ical/), that use several techniques to identify overlapping events and letting the user know about it, in the best cases.

We propose a framework of autonomous robust agents that can automatically help re-schedule appointments, without contravening previous ones, where they can arrange meetings in order to gather people together in a common event at a convenient time.

In an attempt to provide a solid foundation to the agent community, there is an earlier proposal [4] of autonomous intelligent agents to rationally achieve such an autonomy under a changing environment with incomplete information, and little or null human guidance. Despite the nice analysis, it lacks of general results and was founded on a semantics that later presented a couple of issues that jeopardised the integrity of the knowledge bases [5]. Inspired in that work, we present a proposal based on a more-robust theoretical basis for updates and a simple mechanism to reach consensus. As a result, our focus is on preliminary specifications of the inference machine of the involved agents. We claim that the current semantics is appropriate to realize reliable management of the knowledge bases to change specifications, and to achieve a consensual goal.

The following section begins with a background of Answer Sets Programming (ASP hereafter) and updates, where we have founded our proposal, and the update semantics. Section *The Calendar Agent* describes our calendar agents, modelled in a logic programming semantics for updates, where we present several methods to solve conflicting policies and a *result* on specifications change. Next, Section *The General Multi-Agent Setting generalizes* the proposed multi-agent setting. Finally, Section *Conclusions* concludes and describes some future challenges.

BACKGROUND

In this section we briefly introduce some basics about ASP and updates, foundation of our work to get the desired goals.

Intelligent Agents

In order to emphasize a one-to-one relationship between an agent and its user, we use the first person in the following story.

If I had an assistant or secretary who helped me do my job, one of the tasks that I would expect from him/her/*it* is managing my calendar. Although there are already sophisticated systems to help me, I would like to have the possible minimum contact with my calendar, and that my assistant limited itself only to remind me the appointments with enough time in advance, and without overwhelming me. That is to say, the assistant should work for my benefit, be autonomous and awareness, learn, have a rational inference, plan and cooperate with other agents in common goals like a meeting [6, 7, 8].

On the other hand, as gathering people implies negotiating schedules such that they are convenient for the majority, I would like a loyal secretary, always trying to get as much advantage as possible.

To model these features we propose a solid theoretical basis on knowledge representation and reasoning for agent design like belief updates.

Answer-Sets Programming

In this section, we introduce the well-known language of Answer-Sets Programming (or simply ASP) and the class of programs we need is extended disjunctive logic programs (EDLP) as a main foundation of this proposal. Due to space constraints, however, we omit its semantics and language, which can easily be found in the literature. Nevertheless, below we introduce some particular definitions that we need in our context.

Definition 1 ([9]) *An* abductive logic program *is a pair* $\langle \mathcal{P}, \mathcal{A}^* \rangle$ *where* $\mathcal{P}$ *is an arbitrary program and* $\mathcal{A}^*$ *a set of literals, called abducibles.*

Definition 2 (GAS, [9]) *The expression* $\mathcal{M}(\Delta)$ *is a* generalised answer set *of the abductive program* $\langle \mathcal{P}, \mathcal{A}^* \rangle$ *if and only if* $\Delta \subseteq \mathcal{A}^*$ *and* $\mathcal{M}(\Delta)$ *is an answer set of* $\mathcal{P} \cup \{\alpha \leftarrow \top \mid \alpha \in \Delta\}$.

Definition 3 ([10]) *Let* $\mathcal{M}(\Delta_1)$ *and* $\mathcal{M}(\Delta_2)$ *be generalised answer sets of* $\langle \mathcal{P}, \mathcal{A}^* \rangle$. *The relation* $\mathcal{M}(\Delta_1) \leq_{\mathcal{A}^*} \mathcal{M}(\Delta_2)$ *holds if and only if* $\Delta_1 \subseteq \Delta_2$.

Update Semantics

One of the key features of an intelligent agent from the literature is autonomy, which means the agent ability to meet its goals on behalf of the user, by operating on its own without the need of human guidance and by taking the initiative [7]. Accordingly, each agent must have the ability to rearrange its goals as well, by changing its own preferences and policies not to fall into conflict.

The update semantics we employ consists of the following set of definitions taken from [5]:

Formally, an α-*relaxed rule* is a rule ρ that is weakened by a default-negated atom α in its body: $\mathsf{Head}(\rho) \leftarrow \mathsf{Body}(\rho) \cup \{\neg\alpha\}$.

In addition, an *α-relaxed program* is a set of α-relaxed rules. A *generalised program* of $\mathcal{A}^*$ is a set of rules of form $\{l \leftarrow \top \mid l \in \mathcal{A}^*\}$, where $\mathcal{A}^*$ is a given set of literals.

Definition 4 (•-update Program, [5]) *Given an update pair of extended logic programs, denoted as $\mathcal{P}_1 \bullet \mathcal{P}_2$, over a set of atoms $\mathcal{A}$; and a set of unique abducibles $\mathcal{A}^*$, such that $\mathcal{A} \cap \mathcal{A}^* = \emptyset$; and the α-relaxed program $\mathcal{P}'$ from $\mathcal{P}_1$, such that $\alpha \in \mathcal{A}^*$; and the abductive program $\mathcal{P}_{\mathcal{A}^*} = \langle \mathcal{P}' \cup \mathcal{P}_2, \mathcal{A}^* \rangle$. Its* •-update program *is $\mathcal{P}' \cup \mathcal{P}_2 \cup \mathcal{P}_G$, where $\mathcal{P}_G$ is a generalised program of $\mathcal{M} \cap \mathcal{A}^*$ for some* minimal generalised answer set *$\mathcal{M}$ of $\mathcal{P}_{\mathcal{A}^*}$ and "•" is the corresponding update operator.*

Last but not least, the associated *models $\mathcal{S}$ of the new knowledge base* correspond to the answer sets of a •-update program as follows.

Definition 5 (•-update Answer Set, [5]) *Let $\mathcal{P}_\bullet = (\mathcal{P}_1 \bullet \mathcal{P}_2)$ be an update pair of extended logic programs over a set of atoms $\mathcal{A}$. Then, $\mathcal{S} \subseteq \mathcal{A}$ is a* •-answer set of *$\mathcal{P}_\bullet$ if and only if $\mathcal{S} = \mathcal{S}' \cap \mathcal{A}$ for some* minimal generalised answer set *$\mathcal{S}'$ of its •-update program.*

THE CALENDAR AGENTS

Owing to the agent's evolution in time and autonomy, however, we claim that logic programming should be a framework not only for prototyping, but also for a final product as we already lean on reliable efficient solvers from the literature.

Accordingly, the inference machine might be divided into two subparts to make a distinction between both static and dynamic knowledge, as [4] explained. However, as an agent is expected to deal with incomplete information, we claim that even the entire knowledge base should be a object of change.

We illustrate our proposal through an example of three agents and a mediator, later generalised by means of a formal setting. The mediator agent shall be responsible of having a general knowledge base as well as the particular ones from the other agents, always updated.

The Inference Machine

The inference machine is the main basis for this case study encoded into SMODELS [11]. It shall be responsible of managing *beliefs, desires, intentions* and *actions* about its agents and the outside world, according to the literature.

Similar to [4], we introduce three main persistent inertial rules to manage the calendar: *proposal*, the *meeting appointment* itself and *counterproposal*, which are desires, actions and intentions, (*prop*, *meet* and *cprop* in short) respectively, as following illustrated:

```
{meet(1,th,17):- prop(1,a,th,17),
        pref(b,1,17), not app(1,b,th,17),
        pref(c,1,17), not app(1,c,th,17).}
        ⊆ P1
```

The rule represents the following statement: whenever there is a *desire* from agent a to propose meeting 1 on Thursday at 17:00 hr and the slot is free for the rest of the agents b and c, and the proposed hour is in their preferences, *make* it a meeting appointment at that time. Then we shall get an inertial appointment from the corresponding agent.

Scheduling Meeting Appointments

The first step in the process of scheduling an appointment is *the desire*. For example, let us suppose agent a's preference for a meeting is from 13:00 to 18:00 hr, b's from 12:00 to 14:00 hr, and suppose c has no preferences—say, anytime from 8:00 to 18:00. Now suppose a proposes meeting 1 on Thursday at 17:00 hr, represented as $\mathcal{U}_1 = \mathcal{P}_1 \bullet \{\text{prop}(1,a,th,17).\}$ Such an event becomes a desire (a goal) to schedule the meeting. However, the mediator agent, d, concludes that the meeting is impossible just because it falls off b's preferences. So, agent d has to pose a convenient counterproposal.

Counterproposals

In order to figure out the above situation, we have a couple of rules in $\mathcal{P}_1$ that pose a counterproposal when the proposed meeting is impossible:

```
{cProp(1,th,13):- prop(1,a,th,17),
        not meet(1,th,17),
        pref(a,1,13), not app(1,a,th,13),
        pref(b,1,13), not app(1,b,th,13),
        pref(c,1,13), not app(1,c,th,13).}
        ⊆ P1
```

which means that there is a counterproposal say, for Thursday at 13:00 hr, when there was a proposal at 17:00 hr where the meeting was impossible. The counterproposal is based upon the three agents' preferences and their free slots. After the first update, $\mathcal{U}_1$, there is a proposal from a on Thursday at 17:00 hr and a counterproposal from d at 13:00 hr. Next, agent d can derive a list of counterproposals from the corresponding conflicting *desires* at another time, which in turn should be acknowledged by the other agents. Note that counterproposals are agents' capabilities to *negotiate* appointments with the other agents. Several other rules to make counterproposals should be

defined in order to figure out other kinds of conflicts. For the moment, we suppose they exist.

Consensual Meeting

In this framework we propose a *consensus* stage, where each agent has a list of proposals to meet with the other two. First, agent d places a meeting appointment according to the others' preferences. Otherwise, it poses a counterproposal based upon everybody's preferences and waits for acknowledgements. In this example, we have a counterproposal to agent a's proposal. So, agent d has the following set of rules to deal with such situation:

```
{meet(1,th,13):- cProp(1,th,13),
        ack(1,a,th,13), ack(1,b,th,13).
 meet(1,th,13):- cProp(1,th,13),
        ack(1,a,th,13), ack(1,c,th,13).}
        ⊆ P1
```

They mean that a meeting appointment is possible when there is an acknowledged counterproposal from at least one of the other participants and the original proponent: *majority*.

For instance, let us suppose that, from the above counterproposal, all agents acknowledge it. This situation can be easily achieved with a second update:

```
U2 = U1•{ack(1,a,th,13).
        ack(1,b,th,13). ack(1,c,th,13).}
```

Finally we have a *consensual action*, where the agent mediator can already make a meeting appointment on Thursday at 13:00 hr, and everybody is attending.

Changing Specifications

As stated before, we cannot predict all future system specifications, and as a software industry, we would like to automate maintenance for the calendar to meet them, with as much autonomy as possible.

For example, let us recall the possible situation introduced in [12], where several circumstances are not conceived because we cannot anticipate the preferences of all possible users.

- A date after one year from today is considered in the long term.
- Today year is 2011.
- Agent a never schedules long-term appointments.

[2]Some good reasons may be found in the same reference, [4].

Here, the agent states that a long-term appointments never occur.

Now let us suppose somehow the agent states that a long-term appointment is from one year and beyond, and that, for some reason[2] , the user need to schedule an appointment in 2012.

We may represent this situation by means of a set of rules similar to [4], in $\mathcal{P}_1$, although abbreviated as follows.

```
P1 ={   :- app(2012).
        :- app(2013).
        :- app(2014). ...}
```

which means that 2012, 2013, etc. are for some reason (are in the long term) banned. So far, so good.

Next let us suppose the agent updates the calendar with the appointment in 2012. We can model it by $\mathcal{U}_1 = \mathcal{P}_1 \bullet$ {app(2012).}. According to the corresponding update operator, there is an update program

```
:-app(2012), not α1.
:-app(2013), not α2.
:-app(2014), not α3.
app(2012).
```

with a *conflicting rule* (the first one) where the user should decide whether it should be changed or not. Suppose the user does want to change the restriction, by allowing the introduction of α_1 as a fact. As a result, the model corresponds to $\{app(2012), \alpha_1\}$, which means that the appointment in 2012 is no longer banned, and that indeed there is an appointment in such year.

By following the semantics introduced in Section *Background (Update Semantics)*, the rules in the $\bullet$-update program give enough information so as to find the source of conflict, by means of its α-rules, when one of them is satisfied, as shown also in [13].

A rule $\rho \in \mathcal{P}$ is said to *contradict* a consistent EDLP, $\mathcal{P}$, if $\{\rho\} \cup \mathcal{P}$ has no answer sets.

Proposition 1 ([13]) *Suppose two consistent EDLP's $\mathcal{P}_1$, $\mathcal{P}_2$ and the update $\mathcal{P}_1 \bullet \mathcal{P}_2$ with its corresponding α-relaxed program, $\mathcal{P}'_1$, its abductive program $\mathcal{P}_{\mathcal{A}^*}$ with a minimal generalised answer set $\mathcal{M}$; where $\mathcal{P}_1 \cup \mathcal{P}_2$ has no model. The rule $\rho \in \mathcal{P}_1$ contradicts $\mathcal{P}_2$ if and only if its corresponding α-relaxed rule $\rho' \in \mathcal{P}'_1$, where $\alpha \in \mathcal{M}$ and $\neg\alpha \in Body(\rho')$.*

The next step is to ask the user what to do with that appointment and/or the restriction, by means of a meta-program and further development similar

to the counterproposals in the previous section. The user might want to inhibit either, depending on the particular circumstances.

Naturally, there are other trivial cases to consider, like resolving conflicts when making an appointment at the same time of another of lower priority.

So, we have presented preliminary specifications of the inference machine of three calendar agents and a mediator, which can schedule meetings and make counterproposals from conflicting appointments by using a semantics for updates of knowledge bases. Some other proposed features for the society of agents are a consensual stage and changes of specifications. The former consists of a simple majority protocol and the latter of a specifications-change mechanism, which we consider important in an adapting-agent environment.

THE GENERAL MULTI-AGENT SETTING

In order to generalize the story of the consensual meeting, illustrated previous sections, we propose the following framework. So, let us begin with what an agent is.

Definition 6 *Given a knowledge base, Ψ, an agent G is a quintuple (F, G, I, D, R) such that $F, G, I, D, R \subseteq \Psi$ and $F \cap G \cap I \cap D \cap R = \emptyset$. We say that F are facts, G are goals, I are intentions, D are desires and R are restrictions.*

Individually, an agent's knowledge base can be updated following Answer Sets methodologies [5]. In this section, we present a general setting to model agents' agreements. Firstly, we need to establish the communication mechanism that they shall employ.

There are at least two communication mechanisms. The first one consists in a message-passing approach among agents. The second one consists in adding a mediator agent whose purpose is to make the agreement among agents considering their KBs, e.g their beliefs, intentions, etc. We choose the second one since from our point of view is more natural in Answer Sets.

Definition 7 *Let $A_1 = (F_1, G_1, I_1, D_1, R_1), A_2 = (F_2, G_2, I_2, D_2, R_2), \ldots, A_n = (F_n, G_n, I_n, D_n, R_n)$ be agents. A mediator agent is a quintuple (F, G, I, D, R) such that*

$$F = F_1 \cup F_2 \cup \cdots \cup F_n$$
$$G = G_1 \cup G_2 \cup \cdots \cup G_n$$
$$I = I_1 \cup I_2 \cup \cdots \cup I_n$$
$$D = D_1 \cup D_2 \cup \cdots \cup D_n$$
$$R = R_1 \cup R_2 \cup \cdots \cup R_n$$

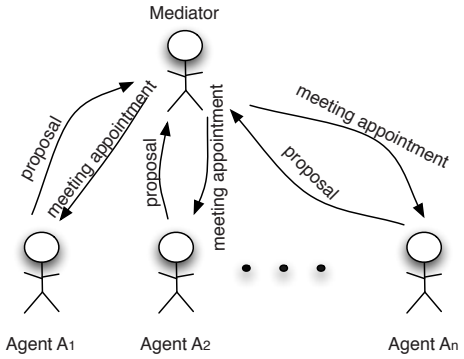

Figure 1 . Multi-agent system

Figure 1 represents the scheme adopted in this work. The mediator keeps an updated copy of the knowledge base of each agent. Each agent can submit a proposal to the mediator to agree an activity.

Definition 8 *A proposal is a request of the form $prop(P, A, D, H)$ where P is an identifier the proposal has, A is the agent that raised the proposal, D and H are the proposed day and time respectively.*

If a proposal is raised by an agent, the mediator shall do the following:

1. Update its KB with the new proposal, i.e include $prop(P)$ to its KB.
2. Find the models of its KB.
   (a) If there is a model in which an agreement of the proposal is factual, then then a meeting is arranged.
   (b) If there are no models in which an agreement of the proposal is present then, there are agents whose schedule does not allow them to agree the proposal. So, the mediator has to find an appropriate counterproposal in order to find models in which the agents can meet. This reasoning seems not appropriate since there should be a consensus among the agents, however, because the mediator has the beliefs, desires and intentions of the agents, the models obtained will consider beliefs, desires and intentions of the agents.
   (c) If there is more than one model in which the counterproposal can be agreeded then, the mediator has to choose one in which most of the agents can meet.

3. Once the meeting has been arranged (dictated by the model obtained), the mediator updates the KB of each agent . The update consists in incorporating the rules from the model, which confirms the meeting on each agent.

Proposition 2 *Let $M = (F, G, I, D, R)$ be a mediator agent, $G_1, G_2, \ldots G_n$ the agents that M mediates and $prop(P, G_i, Day, H)$ a proposal raised by an agent G_i, $0 < i \leq n$. If there is a model of $M' = (F, G, I, D \cup \{S\}, R)$ where a grounding head of the rule S, given by*

$$\begin{aligned} meet(P, Day, H) \quad \leftarrow \\ & prop(P, G_i, Day, H), \\ & pref(G_j, P, H), not\ app(P, G_j, Day, H) \\ & \forall G_j,\ 0 < j \leq n \text{ and } j \neq i \end{aligned}$$

is present then the agent proposal can be confirmed.

Note: The above proposition holds not only in Answer Sets but probably in any other logical framework, since no contradiction arises, e.g there are no pairs of contradictory rules.

Definition 9 *Let $M = (F, G, I, D, R)$ be a mediator agent, $G_1, G_2, \ldots G_n$ the agents that M mediates and $prop(P, G_i, Day, H)$ a proposal raised by an agent G_i, $0 < i \leq n$. A counterproposal is a rule of the form*

$$\begin{aligned} cProp \quad & (P, Day, Hour, \{G_1, G_2, \ldots G_n\}) \leftarrow \\ & prop(P, G_i, Day, H), not\ meet(P, Day, H) \\ & prefer(G_1, P, Day, Hour), \\ & not\ app(G_1, P, Day, Hour), \\ & cProp(P, Day, Hour, \{G_2, \ldots G_n\}) \end{aligned}$$

$$cProp(P, Day, Hour, \{\}) \leftarrow$$

The last fact is a counterproposal raised by the mediator without considering the agents, so it is vacuously true.

Proposition 3 *Let $M = (F, G, I, D, R)$ be a mediator agent, $G_1, G_2, \ldots G_n$ the agents that M mediates and $prop(P, G_i, Day, H)$ a proposal raised by an agent G_i, $0 < i \leq n$. If the head of a counterproposal S is part of the models of $M = (F, G, I, D \cup \{S\}, R)$, then there is a consensus to appoint the desire raised by G_i on the day and time given by Day and H respectively.*

Last, it might happen that not all the agents can agree the desire of G_i, hence the mediator has to find a model in which at least half of the agents.

Definition 10 *Let $M = (F, G, I, D, R)$ be a mediator agent, $G_1, G_2, \ldots G_n$ the agents that M mediates and $cProp(P, Day, H, \{G_1, G_2, \ldots G_n\})$ a counterproposal raised by the mediator. An agreement by majority is a rule of the form*

$$\begin{aligned} meet(P, Day, H) \quad \leftarrow \quad & cProp(P, Day, H, \{G_1, G_2, \ldots G_n\}), \\ & \exists G_1, G_2, \ldots, G_j \in \{G_1, G_2, \ldots G_n\}, \\ & j \geq \lceil n/2 \rceil \\ & ack(P, G_i, Day, H), \forall i\ 0 < i \leq j. \end{aligned}$$

Proposition 4 *Let $M = (F, G, I, D, R)$ be a mediator agent, $G_1, G_2, \ldots G_n$ the agents that M mediates and A an agreement by majority rule. If a grounded head of A is part of a model of $M = (F, G, I \cup \{A\}, D, R)$, then there is a consensus by majority to appoint the counterproposal on the day and time established in the head of A.*

In this section, we have introduced general definitions of the proposed example focused on consensus amongst agents, so that we can have a formal framework to discover further properties and features.

CONCLUSIONS

We present both a case study and generalisation on preliminary specifications of a multi-agent system, and guidelines to a practical example prototype with asynchronous communication amongst intelligent agents. This work provides some evidence to argue how simple and robust an intelligent or rational agent system can be if it is modelled by an appropriate update semantics—syntax independence, fault tolerance and dynamic information. In contrast to [4], we report further development towards a model of negotiation and cooperation on common goals, and suggest simpler but stronger platforms of update languages.

Nevertheless, being still far from a fully-fledged calendar system, much work is to be done. We begin by arguing that updates based on structural properties [5] is a suitable approach for the needs of this an agent society setting, and the examples here introduced can be tested in their implemented solver prototype that is an approximation to the declarative semantics. Such a semantics is founded in ASP, which gives advantages of actual logic programming, efficient solvers, as well as some logics characteristics that provide the agent system with a strong theoretical base to deal with social behaviour.

ACKNOWLEDGMENT

This project has been supported by a project from The Mexican Council of Science and Technology, CONACyT and UAEM.

REFERENCES

[1] Berry, P., Gervasio, M., Peintner, B., Uribe, T., Yorke-Smith, N. (2006). Multi-criteria evaluation in user-centric distributed scheduling agents, in *AAAI Spring Symposium on Distributed Plan and Schedule Management* .

[2] Modi, P.J., Veloso, M., Smith, S. F., Oh, J., (2004). Cmradar: A personal assistant agent for calendar management, in *Agent Oriented Information Systems.* DCR.

[3] Modi, P.J., Veloso, M., (2004). Multiagent meeting scheduling with rescheduling, in *Proceedings of the Fifth Workshop on Distributed Constraint Reasoning.*

[4] Acosta-Guadarrama, J.C., Osorio, M.J., (2003). Towards modelling an intelligent calendar agent with LUPS, in *Applied Informatics,* M. H. Hamza, Ed., IASTED/ACTA Press, 378:60-65.

[5] Acosta-Guadarrama, J.C., (2007). Maintaining knowledge bases at the object level, in *Special Session of the 6th International MICAI Conference,* A. Gelbukh and A. F. Kuri Morales, Eds., pp. 3-13.

[6] Nwana, H.S., Ndumu, D.T., Lee, L.C., Collis, J.C., (1999). ZEUS: a toolkit and approach for building distributed multi-agent systems, in *3rd International Conference on Autonomous Agents (Agents'99)* , O. Etzioni, J. P. Müller, and J. M. Bradshaw, Eds., pp. 360-361.

[7] Nwana, H.S., (1995). Software agents: An overview, *Knowledge Engineering Review,* 11(2):205-244.

[8] Baral, C., Gelfond, M., (1999). Reasoning agents in dynamic domains, in *Workshop on Logic-Based Artificial Intelligence, Washington, DC, June 14--16, 1999* , J. Minker, Ed.College Park, Maryland: Computer Science Department, University of Maryland.

[9] Kakas, A.C., Mancarella, P., (1990). Generalized Stable Models: A semantics for abduction. in *ECAI* , Stockholm, Sweden, pp. 385-391.

[10] Balduccini, M., Gelfond, M., (2003). Logic programs with consistency-restoring rules, in *Proceedings of the AAAI Spring 2003 Symposium* . Palo Alto, California: AAAI Press, pp. 9-18.

[11] Niemela, I., Simons, P., (1997). Smodels---an implementation of the Stable Model and Well-Founded Semantics for normal logic programs, in *Proceedings of the 4th LPNMR ('97)* , LNCS, 1265:420-429.

[12] Acosta-Guadarrama, J.C., Arrazola, J., Osorio, M., (2002). Making belief revision with LUPS, in *XI International Conference on Computing,* J. H. S. Azuela and G. A. Figueroa, Eds.

[13] Acosta-Guadarrama, J.C., Towards a logic-programming system to debug ASP knowledge bases, in *7th Latin American Workshop on Non-Monotonic Reasoning 2011,* M. Osorio, C. Zepeda, I. Olmos, J. L. Carballido, J. Arrazola, and C. Medina, Eds., CEUR Workshop Proceedings, 804:3-12.