A Formal Description of a Practical Agent for E-Commerce

Matteo Pradella and Marco Colombetti

Dipartimento di Elettronica e Informazione,
Politecnico di Milano
P.za Leonardo da Vinci, 32,
20133 Milano, Italia
tel. +39-02-2399-{3666, 3686}
fax. +39-02-2399-3411
{pradella, colombet}@elet.polimi.it

Abstract. Software agents are starting to play a significant role in the field of electronic commerce. Particularly, as mediators they must be completely trusted by the user. This paper tackles this subject by proposing a practically tailored formal model, based on BDI. This model is tested on the ground, firstly, by describing an existing agent for electronic commerce; then, by proving some useful trading properties fulfilled by the specification.

1 Introduction

Software agents are mainly programs to which one can delegate a task. Their quality of (semi)-autonomy is especially useful for the increasingly complex environment of electronic commerce. There is little doubt that software agents will play an increasing variety of roles as mediators in this sensitive field. This roles could be risky: think about an agent delegated to buy, e.g., a car. Somehow a delegating person needs to be confident about the automated agent's correctness. This should be an effective stimulus towards the creation and spreading of a truly automated electronic market. It's a strong motivation for the use of clearly written formal specifications for software agents: agents with great powers should be carefully designed and analyzed.

A formal language is a fundamental tool to describe and analyze the behavior of a software agent. So far some well-founded and interesting formal languages have been proposed, suited to describe and to reason about fairly complex agent environments.

This work consists of a practical application of one of these languages and formal tools in the field of agent-based e-commerce. We used one of the most promising of these languages to specify a practical, implemented software agent, namely a variant of a MIT's Kasbah-based selling agent. We then simplified the chosen language and the chosen set of axioms, just to cover what is needed by our agent.

Kasbah turned to be a quite interesting testing ground: in fact it is one of the more interesting implemented software agent environment for agent-mediated e-commerce (see [7, 9, 6, 5] for recent surveys of the field).

2 Kasbah

Kasbah [2] is a web-based classified ads system, created at MIT. It implements a virtual marketplace with two classes of autonomous agents: *sellers* and *buyers*.

A selling agent is very like to a classified ad. When users create a new selling agent, they give it a description of the item they want to sell. Unlike traditional classified ads, Kasbah selling agents are active. Basically, they try to sell themselves, contacting interested parties (namely, buying agents) and negotiating with them to find the best deal.

Users have some control over the agent's negotiation strategy. They can specify the decay function the agent uses to lower the asking price over its given time frame. The three decay functions offered by Kasbah are linear, quadratic, and cubic with respect to time.

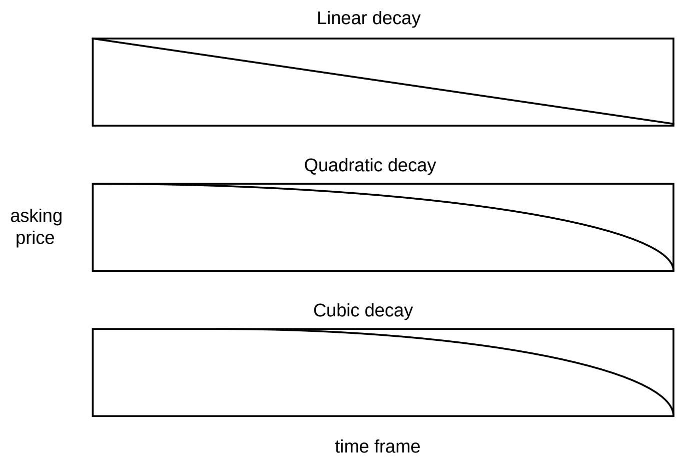

Fig. 1. The three decay functions

To summarize, a Kasbah agent, along with its being a seller or a buyer, is denoted by the following characteristics:

- the good to sell/buy;
- latest desired date to sell/buy the good;
- desired (starting) price;
- lowest/highest acceptable price;
- negotiation strategy (linear/quadratic/cubic with respect to time);

– minimum acceptable reputation of the buyer/seller.

In this work, we will consider only a buyer, without any loss of generality: in fact, a seller is a kind of mirror image of a buyer.

At the time of writing, an improved version of Kasbah is available online, renamed *MarketMaker*, (see `http://maker.media.mit.edu`), which actually introduces the last item of the above dotted list, i.e. the "reputation" management.

3 A stripped-down BDI model

BDI logics is based on a philosophical theory of intentional action originally stated by Bratman (see [1, 11, 12, 10]). It models an autonomous agent, structuring its mental state using three main components: *Belief*, *Desire* (or *Goal*), and *Intention*.

For our specification, we used a modified version of the BDI logic presented in [11, 12]. In fact the specified Kasbah seller turned out to be quite simple; therefore we discarded the D and I components.

Moreover, we never used the branching time Computation Tree Logic (or CTL* - see [3]), as Rao and Georgeff did, and we decided to use a somewhat simpler linear time logic to cover the main timing aspects (see, e.g., the thorough description of Temporal Logic of Concurrency in [4]).

Informally, our semantics works as follows. A *world* is modeled using a linear temporal structure, with single past and single future. A particular time point in a particular world is called a *situation*.

Events transform one time point into another. Of course, the agent may attempt to execute some event, but fail to do so. Thus we distinguish between the successful execution of events and their failure.

As for the timing aspects, as we stated before, we use a version of the Temporal Logic of Concurrency. The standard temporal operator of this logic are: $\bigcirc$, meaning "in the next time instant"; $\Diamond$, meaning "eventually"; $\Box$, meaning "always in the future"; $\mathcal{U}$, meaning "until".

Belief is modeled in a quite conventional way. That is, to each situation, we associate a set of *belief-accessible* worlds: these are, intuitively, these are the worlds that the agent *believes* to be possible (relation $\mathcal{B}$).

The *Intention-accessible* worlds and the *Goal-accessible* worlds of the full BDI model are not considered in this paper.

Let us now enter into some formal details. The following definitions are from [11], slightly modified and tailored to our case.

Definition 1. *An* interpretation M *is a tuple* $M = \langle W, E, T, \prec, U, \mathcal{B}, \Phi \rangle$. W *is a set of worlds,* E *is a set of primitive event types,* T *is a set of time points,* $\prec$ *is a total order relation on* T *(linear time),* U *is the universe of discourse, and* Φ *is a mapping of first-order entities to elements in* U *for any given world and time point. A* situation *is a world, e.g.* w*, at a particular time point, e.g.* t*, and is denoted by* w_t*. The relation* $\mathcal{B}$ *map the agent's current situation to its belief-accessible worlds:* $\mathcal{B} \subseteq W \times T \times W$*. We will use* $\mathcal{B}_t^w$ *to denote the set of worlds* $\mathcal{B}$*-accessible from world* w *at time* t*.*

Definition 2. *Each world w of W is a tuple $\langle T_w, \mathcal{S}_w, \mathcal{F}_w \rangle$, where $T_w \subseteq T$. A* fullpath *is an infinite sequence of time points $(t_0, t_1, \ldots)$ such that $t_i \prec t_{i+1}$. We will use the notation $(w_{t_0}, \ldots)$ to make the world of a particular fullpath explicit. $\mathcal{S}_w : T_w \times T_w \to E$ and similarly of $\mathcal{F}_w$, moreover both are partially functional and have disjoint domains.*

Consider an interpretation M, with a variable assignment v. We take v_d^i to be that function that yields d for the variable i and is the same as v everywhere else. The semantics of first-order formulas can be given as follows.

- $M, v, (w_{t_0}, w_{t_1}, \ldots) \models q(y_1, \ldots, y_n)$ iff $\langle v(y_1), \ldots, v(y_n) \rangle \in \Phi(q, w, t_0)$ where $q(y_1, \ldots, y_n)$ is a predicate formula.
- $M, v, (w_{t_0}, w_{t_1}, \ldots) \models \neg\phi$ iff $M, v, (w_{t_0}, w_{t_1}, \ldots) \not\models \phi$
- $M, v, (w_{t_0}, w_{t_1}, \ldots) \models \phi_1 \vee \phi_2$ iff $M, v, (w_{t_0}, w_{t_1}, \ldots) \models \phi_1$ or $M, v, (w_{t_0}, w_{t_1}, \ldots) \models \phi_2$
- $M, v, (w_{t_0}, w_{t_1}, \ldots) \models \exists i \phi$ iff $M, v_d^i, (w_{t_0}, w_{t_1}, \ldots) \models \phi$, for some d in U
- $M, v, (w_{t_0}, w_{t_1}, \ldots) \models \bigcirc\phi$ iff $M, v, (w_{t_1}, \ldots) \models \phi$
- $M, v, (w_{t_0}, w_{t_1}, \ldots) \models \phi_1 \mathcal{U} \phi_2$ iff $\exists k, k \geq 0$, such that $M, v, (w_{t_k}, \ldots) \models \phi_2$ and $\forall j, 0 \leq j < k, M, v, (w_{t_j}, \ldots) \models \phi_1$
- $M, v, (w_{t_0}, w_{t_1}, \ldots) \models \mathrm{BEL}(\phi)$ iff $\forall w' \in \mathcal{B}_{t_0}^w, M, v, w'_{t_0} \models \phi$
- $M, v, (w_{t_1}, \ldots) \models \mathit{succeeded}(e)$ iff $\exists t_0$ such that $\mathcal{S}_w(t_0, t_1) = e$
- $M, v, (w_{t_1}, \ldots) \models \mathit{failed}(e)$ iff $\exists t_0$ such that $\mathcal{F}_w(t_0, t_1) = e$

Other useful operators are derived as usual:

- $\phi_1 \wedge \phi_2 = \neg(\neg\phi_1 \vee \neg\phi_2)$;
- $\phi_1 \to \phi_2 = \neg\phi_1 \vee \phi_2$;
- $\phi_1 \leftrightarrow \phi_2 = (\phi_1 \to \phi_2) \wedge (\phi_2 \to \phi_1)$;
- $\mathit{true} = (\phi \vee \neg\phi)$;
- $\mathit{false} = \neg \mathit{true}$;
- $\Diamond\phi = \mathit{true}\ \mathcal{U}\ \phi$;
- $\Box\phi = \neg\Diamond\neg\phi$;
- $\exists! i\ p(i) = \exists i(p(i) \wedge \forall j(j \neq i \to \neg p(j)))$.

Here are some of the main definitions about events:

- $\mathit{succeeds}(e) \leftrightarrow \bigcirc \mathit{succeeded}(e)$;
- $\mathit{fails}(e) \leftrightarrow \bigcirc \mathit{failed}(e)$;
- $\mathit{done}(e) \leftrightarrow \mathit{succeeded} \vee \mathit{failed}(e)$;
- $\mathit{does}(e) \leftrightarrow \bigcirc \mathit{done}(e)$.

4 A Kasbah Selling Agent

In this section, we introduce the formal description of our Kasbah-like selling agent. This informal description comes quite directly from [2] and defines the behavior of a Kasbah selling agent.

An agent consists of the following components: control parameters, negotiation history, and internal state. The control parameters are the six user-specified parameters described earlier in the paper. The negotiation history records each conversation that the agent has had with other agents. An example conversation is "I offered agent B a price 100. B rejected the offer" or "Agent C asked my selling price. I replied 91".

Time is partitioned by Kasbah in discrete slices: only one agent is active in a given time slice. The internal state of an agent contains information that the agent uses to decide what will do during its time slice. The internal state stores a list of "potential contacts", which are those agents interested in buying what the agent is selling. With each potential contact, the last known offering price (i.e. what the agent is willing to buy for), and whether it has been asked this round are recorded. The internal state also stores the agent's own current asking price. The strategy an agent uses to decide what to do in each time slice is described below.

- *Current asking price*: the agent lowers its asking price according to the specified price decay function. When the agent is created, its asking price is set to the desired price. By the time to sell by, the asking price is the lowest price. At any moment in between, the current asking price can be interpolated according to the decay function.
- *Decide which agent to talk to*: the agent's strategy is to talk to each potential contact exactly once per round. In other words, an agent will never talk again to a given potential contact until it has first talked to all other potential contacts. The algorithm for deciding which potential contact to talk to during a slice works as follows: consider the potential contacts that have not yet been spoken to in the current round. If all have been spoken to, then begin a new round and consider all the potential contacts. From this set of agents (*suitable*), pick one that has never been contacted, or, if all agents under consideration have been contacted, then pick the one whose last known offering price is the highest. The idea is to first talk to those agents which seem the most promising, first those who have never been spoken to, and then the agents who have indicated a willingness to pay a higher price.
- *Talk to the potential contact* (candidate): the agent offers to sell the item at its current asking price. If the contacted agent accepts, then the agent's job is done. If the contacted agent rejects the offer, then it is asked what its offering price is. This price is recorded for that potential contact.

4.1 The Price Function

The axioms in this section describe the temporal behavior of the offering price function: *price* is a time-dependent function, local to the agent. The specifier needs to include one of the following propositions, depending on the chosen behavior of the price functions: *linearSeller, quadraticSeller, cubicSeller.* These are the only curves available in Kasbah, but other behaviors can be easily introduced by adding a price definition with the chosen function shape.

$dp = max_p - min_p$ and $dt = max_t - min_t$ are agent-dependent constants: i.e. the maximum price variation, and the maximum expected life of the agent, respectively.

$$linearSeller \leftrightarrow \bigwedge_{0 \leq k \leq dt} \Big(start \leftrightarrow \bigcirc^k (price = max_p - \frac{dp}{dt} k) \Big)$$

$$quadraticSeller \leftrightarrow \bigwedge_{0 \leq k \leq dt} \Big(start \leftrightarrow \bigcirc^k (price = max_p - \frac{dp}{dt^2} k^2) \Big)$$

$$cubicSeller \leftrightarrow \bigwedge_{0 \leq k \leq dt} \Big(start \leftrightarrow \bigcirc^k (price = max_p - \frac{dp}{dt^3} k^3) \Big)$$

The following are the corresponding price functions for the "mirror image": a buying agent.

$$linearBuyer \leftrightarrow \bigwedge_{0 \leq k \leq dt} \Big(start \leftrightarrow \bigcirc^k (price = min_p + \frac{dp}{dt} k) \Big)$$

$$quadraticBuyer \leftrightarrow \bigwedge_{0 \leq k \leq dt} \Big(start \leftrightarrow \bigcirc^k (price = min_p + \frac{dp}{dt^2} k^2) \Big)$$

$$cubicBuyer \leftrightarrow \bigwedge_{0 \leq k \leq dt} \Big(start \leftrightarrow \bigcirc^k (price = min_p + \frac{dp}{dt^3} k^3) \Big)$$

These axioms permit a very easy way of coding the price function in the specification. Ideally, using the same structure it is possible to describe whatever price function in a pointwise fashion, thus extending the fixed Kasbah behaviors.

4.2 Actions and Predicates

Let Ag and Pr be two distinct subsets of U, let $a \in Ag$ be an agent, and $p \in Pr$ a price. The following are the main possible actions:

- wants(a,p): a offers the price p for the good;
- info(a,p): a asks the agent its current price, the agent replies p;
- offer(a,p): the agent offers a to buy at price p;
- ask(a,p): the agent asks to a its price, a answers p.

Note that a successful transaction is modeled using *succeeded*. For example, $succeededoffer(a, p)$ means that during the previous time instant a offered a price p, and the agent accepted.

The following are needed predicates about the other agents.

– suitable(a): a is a suitable agent (i.e. it is looking for what we are trying to sell and it has an acceptable reputation);
– dead(a): the agent a is dead (in fact, it ended its activity).

These predicates are "local" to the agent. They are needed to record every negotiation phase and to denote the staring (and ending) of activity.

– candidate(a): a is chosen for negotiation;
– db(a,p): database with agent a and associated price p;
– nc(a): agent a has never been contacted;
– start: the agent is starting its activity;
– end: the agent finished its dt time slot, or successfully sold its good: end of activity.

4.3 Main Axioms

The following axioms define the start and the end of the agent's trade activity (ld1,2,4,5); while (ld3) states about other agents' ending of activity (in fact $dead(a)$ means only that a is no more active for trading). Axiom (ld6) tells about actions and time slice: the agent can execute only one action per round.

(ld1) $start \to \bigcirc\Box\neg start$
(ld2) $start \to \bigcirc^{dt}\Box end \wedge \neg end\,\mathcal{U}\, end$
(ld3) $dead(a) \to \Box dead(a)$
(ld4) $end \to \Box end$
(ld5) $end \to \neg\exists a\exists p(does\,offer(a,p) \vee does\,ask(a,p))$
(ld6) $does\ s \to \neg\exists t(t \neq s \wedge does\ t)$

The next groups of axioms deal with the mental state of the agent.

A first, necessary axiom is based on the previously introduced price function axioms, e.g. if we want to define a linear seller, then we must introduce the axiom BEL($linearSeller$).

Negotiation Management These axioms deal about asking and offering prices for the good. The agent must contact every suitable - and not contacted-before - candidate; while it must offer to candidate and contacted-before agents a price depending to their negotiation history.

(nm1) BEL$(candidate(a) \wedge nc(a) \wedge \neg failed\,offer(a,p)) \to$
$does\,offer(a, price)$
(nm2) BEL$(candidate(a) \wedge nc(a) \wedge failed\,offer(a,p)) \to$
$does\,ask(a,q)$
(nm3) BEL$(candidate(a) \wedge db(a,p) \wedge p \leq price) \to\ does\,offer(a,price)$
(nm4) BEL$(candidate(a) \wedge db(a,p) \wedge p > price) \to\ does\,offer(a,p)$
(nm5) BEL$(done\,wants(a,p) \wedge p \geq price) \to succeeded\,wants(a,p)$
(nm6) BEL$(\exists a\exists p(succeeded\,offer(a,p) \vee succeeded\,wants(a,p))) \to end$

Negotiation recording These axioms define how to insert new entries in the negotiation management database. The agent must remember every highest offer coming from buying agents; moreover it must hold every useful db entry (i.e. concerning only "alive" agents).

(nr1) $(\mathrm{BEL}(done\, wants(a,p)) \vee \mathrm{BEL}(done\, ask(a,p))) \rightarrow \mathrm{BEL}(db(a,p))$
(nr2) $\mathrm{BEL}(db(a,p) \wedge \neg\exists q(done\, wants(a,q)) \wedge \neg\exists q(done\, ask(a,q))) \rightarrow$
$\mathrm{BEL}(\bigcirc(db(a,p) \vee dead(a)))$

DB Consistency The following axioms are basically needed to maintain the negotiation database consistency.

(db1) $\mathrm{BEL}(start \wedge suitable(a) \rightarrow nc(a))$
(db2) $\mathrm{BEL}(db(a,p) \rightarrow \neg nc(a))$
(db3) $\mathrm{BEL}(nc(a) \rightarrow \neg\exists p(db(a,p)))$
(db4) $\mathrm{BEL}(dead(a) \rightarrow \neg nc(a) \wedge \neg\exists p(db(a,p)))$
(db5) $\mathrm{BEL}(suitable(a) \rightarrow (nc(a) \vee \exists! p(db(a,p))))$
(db6) $\mathrm{BEL}((nc(a) \vee \exists p(db(a,p))) \rightarrow suitable(a))$
(db7) $\mathrm{BEL}(\neg dead(a) \rightarrow (nc(a) \rightarrow suitable(a)))$
(db8) $\mathrm{BEL}(\neg dead(a) \rightarrow (\exists p(db(a,p)) \rightarrow suitable(a)))$
(db9) $\mathrm{BEL}(suitable(a) \rightarrow \bigcirc(suitable(a) \vee dead(a)))$

Candidate Management These axioms describes how a candidate is chosen by the agent, and the relation between a candidate and a suitable agent.

(cm1) $\mathrm{BEL}(candidate(a) \wedge db(a,p) \rightarrow$
$\neg\exists b(b \neq a \wedge suitable(b) \wedge db(b,q) \wedge q > p))$
(cm2) $\mathrm{BEL}(candidate(a) \wedge db(a,p) \rightarrow \neg\exists b(b \neq a \wedge nc(b)))$
(cm3) $\mathrm{BEL}(candidate(a) \wedge nc(a) \rightarrow$
$\neg\exists b(b \neq a \wedge nc(b) \wedge failed\, offer(b,p)))$
(cm4) $\mathrm{BEL}(\exists! a(suitable(a) \wedge candidate(a)))$
(cm5) $\mathrm{BEL}(candidate(a) \rightarrow suitable(a))$
(cm6) $\mathrm{BEL}(succeeded\ wants(a,p) \rightarrow \neg\exists b(candidate(b)))$
(cm7) $\mathrm{BEL}(candidate(a) \rightarrow \neg\exists b(b \neq a \wedge does\ offer(b,p)))$

4.4 Cognitive Architecture Axioms

In this section we introduce the cognitive architecture of our agent. Indeed, the architecture turns out to be quite simple: the agent is purely reactive.

Time The agent must hold its belief in the immediate future.

(tm) $\mathrm{BEL}(\bigcirc t) \rightarrow \bigcirc(\mathrm{BEL}(t))$

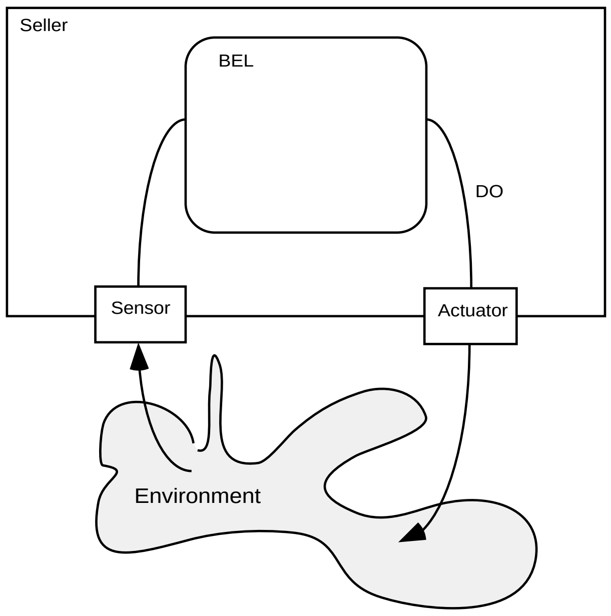

Fig. 2. The agent's cognitive architecture

Sensors Let e be an observable action. The following axioms show how environment affects the agent's "inner belief".

Clearly we use the term "sensors" speaking of a software: in this case the environment is the Kasbah virtual marketplace.

(se1) $\mathit{done}(e) \rightarrow \mathrm{BEL}(\mathit{done}(e))$
(se2) $\mathit{suitable}(a) \rightarrow \mathrm{BEL}(\mathit{suitable}(a))$
(se3) $\mathit{dead}(a) \rightarrow \mathrm{BEL}(\mathit{dead}(a))$
(se4) $\mathit{start} \rightarrow \mathrm{BEL}(\mathit{start})$
(se5) $\mathit{end} \rightarrow \mathrm{BEL}(\mathit{end})$

Actuators Originally we used the following axiom to describe the "virtual-actuator":

(ac) $\mathrm{INTEND}(\mathit{does}(e)) \rightarrow \mathit{does}(e)$

Now, this uses the INTEND operator, but this modal operator was previously present only in this simple form: $\mathrm{INTEND}(\mathit{does}(e))$.

Actually, it is convenient to replace this subformula with $\mathit{does}(e)$, because the INTEND part is totally absent from this agent's architecture (as well as the GOAL part). Therefore (*ac*) can be deleted. Clearly, as was stated before, this is a purely reactive agent, and therefore the concept of intention is redundant.

About Axiom Systems Rao and Georgeff preferred axiom system for the BEL component is KD45 (see [12]). In our example, the BEL component is in fact functioning like a simple database for the agent. We need basic logic deduction within this BEL component (K). We want belief to be consistent (D), but we do not need introspection and there are not nested applications of the BEL operator: we can forget axioms like (4) or (5). Therefore, we use just the following axioms:

(K) $\mathrm{BEL}(\phi) \wedge \mathrm{BEL}(\phi \rightarrow \psi) \rightarrow \mathrm{BEL}(\psi)$
(D) $\mathrm{BEL}(\phi) \rightarrow \neg\mathrm{BEL}(\neg\phi)$

Another relevant aspect is *time*: considering the CTL component, we do not need a branching time logic, because all time operators have the form *inevitable OP*. In our opinion, branching time logic is effective only with more sophisticated agents, able to deal with quite complex situations in a deliberative way. Therefore linear temporal logic suffices, allowing us to drop the *inevitable* operator.

5 Some Useful Trading Properties

As we stated in the introduction, the user may want to be assured about the correctness of his agent. Now, having a formal definition of it, we can assert -and prove- useful properties.

A first interesting property is the following: the agent will never offer a price below the price function. This is stated by the next theorem.

Theorem 1. $BEL(candidate(a)) \wedge does(offer(a,p)) \rightarrow p \geq price$

Proof. We will partition the scenario in mutual exclusive cases, depending on the presence/absence of the entry referred to agent a in the agent's database (see axioms (db$n$)).

Let us suppose that $db(a,p')$, with $p' \leq price$. Then, by (mn3), it implies $does(offer(a,price))$. But, by (ld6), we cannot ask/receive other offers. Therefore $p = price$: the statement holds.

Now suppose $db(a,p')$, with $p' > price$.

Then, by (mn4), this implies $does(offer(a,p'))$. Like in the previous case, by (ld6), we cannot ask or receive other offers. Therefore $p = p' > price$: the statement holds.

Consider now the case $nc(a)$. We can suppose $done(offer(a,p'))$ (case a) and specifically succeeded $offer(a,p')$ (case a1). This implies, by (nm6), *end*, therefore (by axiom (ld5)): $\neg\exists a\exists p(does(offer(a,p)))$. The premise of the statement is false: the statement holds.

Otherwise, we can suppose $failed(offer(a,p'))$ (case a2). This implies (nm2) $does(ask(a,p'))$, therefore we have (ld7) $\neg\exists a\exists p(does(offer(a,p)))$. Like in the previous case, the premise of the statement is false: the statement holds.

Now consider (case b): $\neg done(offer(a,p'))$. This implies (definition of *done*): $\neg failed(offer(a,p'))$.

Therefore, by (nm1), we obtain $does(offer(a,price))$, but, by (ld6), we cannot ask/receive other offers. Therefore $p = price$: the statement holds.

The following other simple statement assures that the agent, after receiving an offer higher or equal to the current *price*, will buy immediately. Moreover, it will not offer/ask anymore (*end of activity*).

Theorem 2. $done(wants(a,p)) \wedge p \geq price \rightarrow$
$succeeded(wants(a,p)) \wedge \neg\exists b(does(offer(b,q)))$

Proof. By (se1), the premise of the statement implies the premise of axiom (nm5). Therefore succeeded $wants(a,p)$ holds. But this, by (nm6), implies the end of activity (*end*).

This means, using (ld5), that $\neg\exists b(does(offer(b,q)))$ holds, which is the other consequence to be proved.

Actually, the previous two properties are fairly trivial but nonetheless useful. They are thought just as an example, to show how this can be an effective framework for asserting trading properties in the delicate field of e-commerce.

As stated in [8], a practical formal method should be supported by analysis techniques that can be invoked with the mere push of a button (e.g. model checking). A really *simple* formal framework, like the one presented in this paper, is naturally suited to "pushbutton" techniques.

6 Conclusions

First generation systems for agent-mediated electronic commerce are already creating new markets and beginning to reduce transaction costs in a variety of business tasks. However, many business aspects are now only partially (if any) covered, and sometimes in an ineffective or unreliable way. A more extensive use of formal methods should change and improve the current situation.

For this point, this experience turned to be quite useful in some respects.

We effectively use some well-known formal tools to *reverse-engineer* a real implemented e-market autonomous software agent. This naturally permits to proof useful properties about the correctness of the agent's behavior.

We simplified the BDI formalism in a somewhat weaker, but easier and more practical-suited logic. Quite naturally, an "easy" logic usually means easier deductions and the possibility of efficient (semi-)automatic analysis tools.

We did a simple and straightforward reasoning about the cognitive architecture of our agent. It seems quite natural that a practical agent for e-commerce must be simple and straight-minded: network bandwidth and host processing power are at present too weak to adequately support the refined and complex-minded agents we often see in some theoretical/philosophical papers. As a matter of fact, sometimes the expressive power of the chosen logic results misdirected, compared to the requirements of present (and probably near-future) automated electronic market.

7 Acknowledgments

We wish to thank Gianpaolo Cugola and Stefano Gaburri for the fruitful discussions. Last but not least, many thanks even to Mattia Monga, who solved some nontrivial LaTeX formatting problems.

References

1. M. E. Bratman. *Intentions, Plans, and Practical Reason.* Harvard University Press, Cambridge, MA, 1987.
2. A. Chavez and P. Maes. Kasbah: An agent marketplace for buying and selling goods. In *First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology*, pages 75–90, 1996.
3. E. Allen Emerson and Jai Srinivasan. Branching time temporal logic. In Jaco W. de Bakker, Willem-Paul de Roever, and Grzegorz Rozenberg, editors, *Linear Time, Branching Time and Partial Order in Logics and Models of Concurrency*, number vol. 354 in Lecture Notes in Computer Science, pages 123–172. Springer, Heidelberg, Germany, 1989.
4. R. Goldblatt. *Logics of Time and Computation, Second Edition, Revised and Expanded*, volume 7 of *CSLI Lecture Notes.* CSLI, Stanford, 1992 (first edition 1987). Distributed by University of Chicago Press.
5. R. H. Guttman and P. Maes. Cooperative vs. competitive multi-agent negotiations in retail electronic commerce. In Matthias Klusch and Gerhard Weiß, editors, *Proceedings of the 2nd International Workshop on Cooperative Information Agents II: Learning, Mobility and Electronic Commerce for Information Discovery on the Internet*, volume 1435 of *LNAI*, pages 135–147, Berlin, July 4–7 1998. Springer.
6. R. H. Guttman and P. Maes. Agent-mediated integrative negotiation for retail electronic commerce. In Pablo Noriega and Charles Sierra, editors, *Proceedings of the 1st International Workshop on Agent Mediated Electronic Commerce (AMET-98)*, volume 1571 of *LNAI*, pages 70–90, Berlin, May 10–10 1999. Springer.
7. R. H. Guttman, A. G. Moukas, and P. Maes. Agent-mediated electronic commerce: A survey. In *Knowledge Engineering Review*, 1998.
8. C. Heitmeyer. On the need for practical formal methods. *Lecture Notes in Computer Science*, 1486, 1998.
9. P. Maes, R. H. Guttman, and A. G. Moukas. Agents that buy and sell. *Communications of the ACM*, 42(3):81–91, March 1999.
10. D. Morley. Semantics of BDI agents and their environment. Technical Report 74, Australian AI Institute, Carlton, Australia, 1996.
11. A. S. Rao and M. P. Georgeff. Modeling rational agents within a BDI-architecture. Technical Report 14, Australian AI Institute, Carlton, Australia, 1991.
12. A. S. Rao and M. P. Georgeff. Formal models and decision procedures for multi-agent systems. Technical Note 61, Australian Artificial Intelligence Institute, Melbourne, Australia, June 1995.