Agent programming with priorities and deadlines

Konstantin Vikhorev Natasha Alechina Brian Logan
School of Computer Science
University of Nottingham
Nottingham NG8 1BB, UK
{kxv,nza,bsl}@cs.nott.ac.uk

ABSTRACT

We present AgentSpeak(RT), a real-time BDI agent programming language based on AgentSpeak(L). AgentSpeak(RT) extends AgentSpeak intentions with *deadlines* which specify the time by which the agent should respond to an event, and *priorities* which specify the relative importance of responding to a particular event. The AgentSpeak(RT) interpreter commits to a priority-maximal set of intentions: a set of intentions which is maximally feasible while preferring higher priority intentions. We prove some properties of the language, such as guaranteed reactivity delay of the AgentSpeak(RT) interpreter and probabilistic guarantees of successful execution of intentions by their deadlines.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: Programming Languages and Software

General Terms

Languages, Theory

Keywords

Agent programming languages, Belief Desire and Intention logics, Complexity of reasoning

1. INTRODUCTION

Belief-Desire-Intention (BDI) based agent programming languages facilitate the development of rational agents specified in terms of beliefs, goals and plans. They allow an agent to balance deliberating about which plan to adopt in response to events (changes in its beliefs or goals) and executing its current intentions. An agent is *rational* if it adopts and executes intentions which achieve its goals, given its current beliefs.

If an agent's task environment is *real-time*, the requirements for rational behaviour are more complex. In a real-time environment, the events to which the agent must respond are characterized by a *deadline*, e.g., the time by which a goal must be achieved or the agent must respond to a change in its beliefs. In such an environment, a rational agent should not adopt an intention which it believes cannot be successfully executed by its deadline or continue to execute an intention after its deadline. For example, an agent should not adopt an intention of writing a research proposal which must be submitted by 4pm on Friday if there is insufficient time to write the proposal. Similarly, if the agent believes it cannot achieve every goal or respond to every change in its environment by the relevant deadline, it should adopt intentions for the highest priority events which are feasible.

We define a *real-time BDI agent* as one which is rational in this sense, i.e., it adopts and schedules intentions so as to respond to events by their deadlines; if not all events can be processed by their deadlines, the agent favours intentions responding to high priority events. For a real-time BDI agent, correctness of the agent's program depends not only on the actions the agent performs but the time at which it performs them. However programming real-time BDI agents in most existing BDI languages is hard as they lack the notion of a deadline or the ability to deliberate about the feasibility of intentions.

In this paper we present AgentSpeak(RT), a programming language for real-time BDI agents in applications such as UAVs, process control, trading agents, etc. AgentSpeak(RT) extends AgentSpeak(L) [9] with deadlines and priorities, and, given the estimated execution time of plans, schedules intentions so as to achieve a priority-maximal set of intentions by their deadlines with a specified level of confidence. Real-time tasks can be freely mixed with tasks for which no deadline and/or priority has been specified, and if no deadlines and priorities are specified, the behaviour of the agent defaults to that of a non real-time BDI agent. We prove a number of properties of AgentSpeak(RT), including that the reactivity delay of an AgentSpeak(RT) agent is bounded, that it commits to a priority-maximal set of intentions, and that in a static environment its intentions will complete successfully by their deadlines with specified confidence. We also develop a model of the 'difficulty' of the agent's environment, and show how it can be used to determine the priority of intentions which will complete successfully by their deadlines with specified confidence. A key contribution of the paper is the analysis of real-time guarantees for BDI agents and how these can be achieved within a BDI programming framework. Although our approach to real-time BDI agents is developed in the context of a particular BDI agent programming language, we believe it can be applied to other BDI-based agent programming languages.

The remainder of this paper is organised as follows. In section 2 we present the syntax of AgentSpeak(RT) and briefly describe the execution cycle of the AgentSpeak(RT) architecture. In section 3 we show that under certain reasonable assumptions, the time required to execute a single cycle of the AgentSpeak(RT) interpreter (and hence the reactivity delay of the agent) is bounded. We also prove that an AgentSpeak(RT) agent commits to a priority-maximal set of intentions, and that in a static environment its intentions will

Cite as: Agent programming with priorities and deadlines, Vikhorev, Alechina and Logan, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 397-404.

complete successfully by their deadlines with specified confidence. In section 4 we develop a model of the 'dynamism' of the agent's environment, and show how it can be used to determine the priority of intentions which can be reliably scheduled, and to estimate the probability that a scheduled intention of given priority will be displaced from the schedule by the arrival of an intention of higher priority. Finally, in sections 5 and 6 we briefly review related work and conclude.

2. THE AgentSpeak(RT) ARCHITECTURE

In this section we introduce the AgentSpeak(RT) agent programming language and its associated interpreter.

We assume that an AgentSpeak(RT) agent operates in a real-time task environment in which events (external goals and changes in the agent's beliefs about its environment) may be associated with a deadline and/or a priority. The agent responds to events by adopting and executing intentions. A developer can specify a required level of confidence for the successful execution of intentions in terms of a probability, α, and the agent schedules its intentions so as to ensure that the probability that intentions complete by their deadlines is at least α.[1] Setting $\alpha = 1$ gives hard real-time behaviour, i.e., the agent will only commit to intentions that are certain to complete by their deadlines. If not all intentions can be executed with the required level of confidence, the agent favours intentions triggered by high priority events.

The syntax and semantics of AgentSpeak(RT) is based on AgentSpeak(L) [9]. We briefly review the syntax of AgentSpeak(L) and explain the extensions to support real-time applications. To illustrate the syntax of AgentSpeak(RT) we use a simple running example of a trading agent which buys commodities in an electronic marketplace. The agent receives requests from clients to bid on their behalf, and notifications of goods for sale which the agent may also bid for on its own behalf. The deadline of an event is the deadline for the corresponding auction. The priority is determined by the importance of the client and the type of goods for sale.

The AgentSpeak(RT) architecture consists of five main components: a belief base, a set of events, a plan library, an intention structure, and an interpreter.

2.1 Beliefs and Goals

The agent's beliefs represent its information about its environment, e.g., sensory input, information about other agents, etc. Beliefs are represented as ground atomic formulas. For example, the agent may believe that client1 is a client, and that good1 is the type of good that the agent buys:

```
client(client1)
buys(good1)
```

A belief atom or its negation is referred to as a *belief literal*. A ground belief atom is called a *base belief*, and the agent's belief base is a conjunction of base beliefs.

A goal is a state the agent wishes to bring about or a query to be evaluated. An achievement goal, written $!g(t_1, \ldots, t_n)$ where $t_i, \ldots, t_n$ are terms, specifies that the agent wishes to achieve a state in which $g(t_1, \ldots, t_n)$ is a true belief. A test goal, written $?g(t_1, \ldots, t_n)$, specifies that the agent wishes to determine if $g(t_1, \ldots, t_n)$ is a true belief. For example, the goals

```
!bid(client1, a101, price1)
```

```
?credit(client1, price1)
```

indicate that the agent should bid price1 in auction a101 on behalf of client1, and determine if client1 has sufficient credit to cover price1.

2.2 Events

Changes in the agent's beliefs or the acquisition of new achievement goals give rise to events. An addition event, denoted by $+$, indicates the addition of a base belief or an achievement goal. A deletion event, denoted by $-$, indicates the retraction of a base belief.[2] We distinguish between internal and external events. An external event is one originating in the agent's environment while internal events result from the execution of the agent's program. As in AgentSpeak(L), all belief change events are external, while goal change events may be external (goals originated by a user or another agent) or internal (subgoals generated by the agent's program in response to an external event).

To allow the specification of real-time tasks, external events may optionally specify a deadline and a priority. A *deadline* specifies the time by which a goal should be achieved or the agent should respond to a change in its beliefs. Deadlines are expressed as real time values in some appropriate units, e.g, a user may specify a deadline for a goal as "4pm on Friday". Deadlines in AgentSpeak(RT) are hard—it is assumed that there is no value in achieving a goal or responding to a belief change after the deadline has passed. A *priority* specifies the relative importance of achieving the goal or responding to a belief change. Priorities define a partial order over events and are expressed as non-negative integer values, with larger values taken to indicate higher priority. For example, the events

```
+!bid(client2, a102, price2)[1010, 15]
+auction(a201, good1)[1060, 10]
```

indicates the acquisition of a goal to bid price2 on behalf of client2 in auction a102 with deadline 1010 and priority 15, and a new belief that good1 is being offered in auction a201, with deadline 1060 and priority 10. By default the deadline is equal to infinity and the priority is equal to zero.

2.3 Plans

Plans specify sequences of actions and subgoals an agent can use to achieve its goals or respond to changes in its beliefs. The head of a plan consists of a triggering event which specifies the kind of event the plan can be used to respond to, and a belief context which specifies the beliefs that must be true for the plan to be applicable. The body of a plan specifies a sequence of actions and (sub)goals to respond to the triggering event.

Actions are the basic operations an agent can perform to change its environment in order to achieve its goals. Actions are denoted by *action symbols* and are written $a(t_1, \ldots, t_n)$ where a is an action symbol and $t_1, \ldots, t_n$ are the (ground) arguments to the action. For example, the action

```
send-bid(a102, price2)
```

will cause the agent to bid price2 in auction a102. Performing an action may result in changes in the agent's beliefs when the action's effects on the environment are sensed at subsequent cycles of the interpreter.

Plans may also contain achievement and test (sub)goals. Achievement subgoals allow an agent to choose a course of action as part of a larger plan on the basis of its current beliefs. An achievement subgoal $!g(t_1, \ldots, t_n)$ gives rise to a internal goal addition event

[1] For simplicity, we assume that α is the same for all intentions; however the real time guarantees we prove in sections 3 and 4 still hold if α is different for different events.

[2] In the interests of brevity, we do not consider goal deletion events.

$+!g(t_1, \ldots, t_n)$ which may in turn trigger subplans at the next execution cycle. Test goals are evaluated against the agent's belief base, possibly binding variables in the plan. For example, the plan

```
+!bid(C, A, P,) : client(C) <-
  ?credit(C, P); send-bid(A, P).
```

is triggered by a request from agent `C` to bid price `P` in auction `A`. If `C` is a client, then the agent will check that the client has sufficient credit and, if so, make the bid on the client's behalf.

The BNF for plans is given below:

plan	::=	*event* ["`:`" *context*] "<−" *body* "`.`"
event	::=	"+" ["`!`"] *atomic-formula* | "−" *atomic-formula*
context	::=	*true* | *literal* ("`&`" *literal*)*
literal	::=	*atomic-formula* | "`not`" *atomic-formula*
atomic-formula	::=	$p(t_1, \ldots, t_n)$
body	::=	*true* | *step* ("`;`" *step*)*
step	::=	$a(t_1, \ldots, t_n)$ | "`!`" *atomic-formula* | "`?`" *atomic-formula*

where p and a are respectively predicate and action symbols of arity $n \geq 0$, and $t_1, \ldots, t_n$ are terms. (As in Prolog, constants are written in lower case and variables in upper case, and all negations must be ground when evaluated.)

2.4 Execution Time Profiles

In order to determine whether a plan can achieve a goal by a deadline with a given level of confidence, each action and plan has an associated *execution time profile* which specifies the probability that the action or plan will terminate successfully as a function of execution time. We assume that plans can be arbitrarily interleaved, and the estimated execution time of a plan is independent of any other plans the agent is currently executing. The expected execution time for an action or plan ϕ at confidence level α is given by $et(\phi, \alpha)$. We assume that execution times increase monotonically with α, i.e., in general, to have higher confidence that a plan will complete successfully, we need to allow more time for the plan to execute. The shape of the execution time profile will typically be influenced by the (assumed) characteristics of the environment in which the agent will operate. For example, the probability of a plan to move to a location terminating successfully within a given time may be lower in environments with many obstacles than in environments with fewer obstacles. Execution time profiles can be derived from an analysis of the agent's actions, plans and environment, or using automated techniques, e.g., stochastic simulation. In the simple case of plans consisting of a sequence of actions, the execution time profile for the plan can computed from the execution time profiles of its constituent actions. However for plans which contain subgoals, the execution time profile will depend on the relative frequency with which the possible plans for a subgoal are selected in the agent's task environment.

2.5 Intentions

The intention structure contains plans that have been chosen to achieve goals or respond to changes in the agent's beliefs. Plans triggered by changes in beliefs or the acquisition of an external (top-level) achievement goal give rise to new intentions. Plans triggered by the processing of an achievement subgoal in an already intended plan are pushed onto the intention containing the subgoal. Each intention consists of a stack of partially executed plans, a set of substitutions for plan variables, and a deadline and priority. The set of variable substitutions for each plan in an intention results from matching the belief context of the plan and any test goals it contains against the agent's belief base. The deadline and priority of an intention are determined by the triggering event of the root plan.

2.6 The AgentSpeak(RT) Interpreter

The interpreter is the main component of the agent. It manipulates the agent's belief base, events and intention structure, deliberates about which plan to select in response to belief and goal change events, and schedules and executes intentions.

The agent's state is a tuple $\langle B, E, I\rangle$ consisting of a set of base beliefs B, a set of events E, and an (ordered) set of intentions I. We formalize the execution of the interpreter as a sequence of function applications which compute the new state of the agent and an executed action based on its current state and its inputs at the current cycle

$$(\langle B', E', I'\rangle, a) = exec(sched(opt(evt(\langle B, E, I\rangle, P, G))))$$

where P is a set of percepts, G is a set of external goal addition events, B', E', I' are the updated belief, event and intention sets, and a is an action or null.

The function evt generates a set of events based on the agent's percepts P and external goal addition events G. It updates the belief base B with the percepts in P to give an updated belief base B' and a set of belief addition and removal events E_P, and returns a new state

$$\langle B', E_1 = E \cup E_P \cup G, I\rangle = evt(\langle B, E, I\rangle, P, G)$$

The second function, opt, takes $\langle B', E_1, I\rangle$ as input and returns a pair consisting of a new state and a set of applicable plans or options O

$$(\langle B', \emptyset, I_1\rangle, O) = opt(\langle B', E_1, I\rangle)$$

In contrast to AgentSpeak(L) which processes a single event at each interpreter cycle, to ensure reactivity, AgentSpeak(RT) iterates through E_1, and, for each event $e \in E_1$, generates a set of applicable plans O_e. A plan is *relevant* if its triggering event can be unified with e and a relevant plan is *applicable* if its belief context is true in B'. In general, there may be many applicable plans or options for each event. A selection function S_O chooses one of these plans for each event to give a set of options $O = \{S_O(O_e) \mid e \in E_1\}$. S_O is a partial function, i.e., it is not defined if O_e is empty. If the event was triggered by a subgoal of an existing intention, failure to find a applicable plan for the subgoal, i.e., if $O_e = \emptyset$, aborts the intention which posted the subgoal and the intention is removed from I (hence the change from I to I_1).

The third function, $sched$, takes $(\langle B', \emptyset, I_1\rangle, O)$ as input and returns a new state

$$\langle B', \emptyset, I_2\rangle = sched(\langle B', \emptyset, I_1\rangle, O)$$

For each plan π in O, it either pushes π on top of the existing intention in I_1 that generated the triggering event (if the triggering event for π was internal), or creates a new intention τ and adds it to a set I_E (if the triggering event for π was external). I_2 is the result of applying the scheduling algorithm (see Algorithm 2 below) to $I_1 \cup I_E$. The scheduling algorithm returns a set of feasible intentions in deadline order (earliest deadline first).

Finally, $exec$ takes $\langle B', \emptyset, I_2\rangle$ as input and returns a pair consisting of a new state and an executed action

$$(\langle B', E', I'\rangle, a) = exec(\langle B', \emptyset, I_2\rangle)$$

where I' is the result of executing the first intention in the schedule I_2, E' contains any internal goal addition event generated by executing the intention, and a is the action executed (or null if no action was executed). Executing an intention involves executing

the first goal or action of the body of the topmost plan in the stack of partially executed plans which forms the intention. Executing an achievement goal adds a corresponding internal goal addition event to E'. Executing a test goal involves finding a unifying substitution for the goal and the agent's base beliefs. If a substitution is found, the test goal is removed from the body of the plan and the substitution is applied to the plan. If no such substitution exists, the test goal is not removed and may be retried at the next cycle. Executing an action results in the invocation of the Java code that implements the action. If the action completes within its expected execution time $et(a, \alpha)$, it is removed from the body of the plan. Actions which time out are not removed and may be retried at the next cycle.[3] The executed action is returned as a. Reaching the end of a plan (denoted by *true* below) causes the plan to be popped from the intention and any substitutions for variables appearing in the head of the popped plan are applied to the topmost plan in the intention.

Algorithm 1 AgentSpeak(RT) Interpreter Cycle

$E := E \cup G \cup \mathit{belief\text{-}events}(B, P)$
$B := \mathit{update\text{-}beliefs}(B, P)$
for all $\langle e, \tau \rangle \in E$ **do**
  $O_e := \{\pi\theta \mid \theta$ is an applicable unifier for e and plan $\pi\}$
  $\pi\theta := S_O(O_e)$
  if $\pi\theta \neq \emptyset$ **and** $\tau \notin I$ **then**
    $I := I \cup \pi\theta$
  else if $\pi\theta \neq \emptyset$ **and** $\tau \in I$ **then**
    $I := (I \setminus \tau) \cup \mathit{push}(\pi\theta\sigma, \tau)$ where σ is an mgu for $\pi\theta$ and τ
  else if $\pi\theta = \emptyset$ **and** $\tau \in I$ **then**
    $I := I \setminus \tau$
  end if
end for
$I :=$ SCHEDULE(I)
if $I \neq \emptyset$ **then**
  $\tau := \mathit{first}(I)$
  if $\mathit{first}(\mathit{body}(\mathit{top}(\tau))) = \mathit{true}$ **then**
    $\pi := \mathit{pop}(\tau), \pi' := \mathit{pop}(\tau)$
    $\mathit{push}((\mathit{head}(\pi') \leftarrow \mathit{rest}(\mathit{body}(\pi')))\theta, \tau)$
    where θ is an mgu such that $\mathit{head}(\pi)\theta = \pi'\theta$
  else if $\mathit{first}(\mathit{body}(\mathit{top}(\tau))) = !g(t_1, \ldots, t_n)$ **then**
    $E = \{\langle +!g(t_1, \ldots, t_n), \tau \rangle\}$
  else if $\mathit{first}(\mathit{body}(\mathit{top}(\tau))) = ?g(t_1, \ldots, t_n)$ **then**
    if $?g(t_1, \ldots, t_n)\theta$ is an answer substitution **then**
      $\pi := \mathit{pop}(\tau)$
      $\mathit{push}((\mathit{head}(\pi) \leftarrow \mathit{rest}(\mathit{body}(\pi)))\theta, \tau)$
    end if
  else if $\mathit{first}(\mathit{body}(\mathit{top}(\tau))) = a(t_1, \ldots, t_n)$ **then**
    if $\mathit{execute}(a(t_1, \ldots, t_n), \mathit{et}(a(t_1, \ldots, t_n), \alpha))$ **then**
      $\pi := \mathit{pop}(\tau)$
      $\mathit{push}(\mathit{head}(\pi) \leftarrow \mathit{rest}(\mathit{body}(\pi)), \tau)$
    end if
  end if
end if

The interpreter code is shown in Algorithm 1. The functions *head* and *body* return the head and body of an intended plan, and *first* and *rest* are used to return the first and all but the first elements of a sequence. The function *top* returns the topmost plan in an intention. The function *pop* removes and returns the topmost plan of an intention and the function *push* takes a plan (and any substitution) and an intention and pushes the plan onto the top of the intention. The function *execute* takes an action and an expected execution time, and executes the action for at most the expected execution time. It returns true if the action completes successfully within its expected execution time; otherwise it returns false.

The scheduling algorithm is shown in Algorithm 2. The set of candidate intentions is processed in descending order of priority.[4] A candidate intention is added to the schedule if it can be inserted into the schedule in deadline order while meeting its own and all currently scheduled deadlines. A set of intentions $\tau_1, \ldots, \tau_n$ is feasible if there exists a schedule where each intention is executed before its deadline. To check whether a schedule exists for a set of intentions ordered earliest deadline first, it suffices to check that for every scheduled intention τ_i

$$\Sigma_{j \leq i}\, et(\tau_j, \alpha) - ex(\tau_j) \leq d(\tau_i)$$

where $ex(\tau_j)$ is the time τ_j has spent executing up to this point, and $d(\tau_i)$ is the deadline for τ_i. That is, the sum of expected remaining execution time of intentions scheduled earlier than τ_i including τ_i itself is less that the deadline of τ_i. A set of tasks is feasible iff they can be scheduled earliest deadline first [7]. Intentions which are not feasible in the context of the current schedule or which have exceeded their expected execution time are dropped.[5]

Algorithm 2 AgentSpeak(RT) Scheduling Algorithm

function SCHEDULE(I)
  $\Gamma := \emptyset$
  for all $\tau \in I$ in descending order of priority **do**
    if $\{\tau\} \cup \Gamma$ is feasible **then**
      $\Gamma := \{\tau\} \cup \Gamma$
    end if
  end for
  sort Γ in order of increasing deadline
  return Γ
end function

The scheduler returns a set of intentions which is 'maximally feasible' (no more intentions can be added to the schedule if the scheduled intentions are to remain feasible at the specified confidence level) and moreover, intentions which are dropped are incompatible with some scheduled higher priority intention(s). Scheduling in AgentSpeak(RT) is pre-emptive in that the adoption of a new high-priority intention τ_i may prevent previously scheduled intentions with priority lower than i (including the currently executing intention) being added to the new schedule.

Note that if deadlines and priorities are not specified for external events (and hence $d = \infty, p = 0$ for all intentions), $et(\phi, \alpha) = \infty$ for all $\phi, 0 \leq \alpha \leq 1$, the behaviour of an AgentSpeak(RT) agent defaults to that of a non real-time BDI agent.

2.7 Implementation

We have implemented AgentSpeak(RT) in Java, and the current prototype implementation includes the core language described above and implementations of some basic actions. Additional user-defined actions can be added using a Java API. AgentSpeak(RT) supports

[3] Allowing test goals and actions to be retried is not critical, but means that successful execution of intentions is less dependent on precise characterization of the execution time profile of actions.

[4] If there are multiple intentions with the same priority and/or deadline, we assume they are processed in a fixed order.

[5] The real time guarantees we prove in section 3 still hold in some circumstances if intentions that exceed their expected execution time are not dropped, but it complicates the presentation. The basic idea is that an intention τ which has exceeded its expected execution time has its priority reduced to 0. τ will only be scheduled if, after scheduling all higher priority intentions, there is sufficient slack in the schedule to execute at least one step in τ before its deadline. Given sufficient slack in the schedule, τ can therefore still complete successfully. It will be however dropped if it exceeds its deadline.

two mechanisms for defining primitive actions: writing a class which implements the `ExternalAction` interface, and direct invocation of methods in existing Java legacy code.

2.8 Example

In this section we sketch a simple example AgentSpeak(RT) agent and show how it allows the specification of tasks with priorities and deadlines.

Consider a trading agent which buys commodities in an electronic marketplace both on its own behalf and as a broker on behalf of clients. The market operates as a series of concurrent first-price sealed-bid auctions of short duration in which sellers offer goods for sale. Each auction has a deadline by which bids must be received. Once the deadline for an auction has passed, the market determines the highest bid and notifies successful agents of their purchase and remaining credit level. The trading agent responds to two kinds of events: requests from clients to make a specified bid on their behalf in a particular auction, and notifications of new auctions where the agent may decide to bid on its own behalf. The deadline of an event is the deadline for bids for the corresponding auction, and priorities are assigned to events based on the importance of the client (for client requests) and the type good sold in the auction (for auction notifications). The agent's primary role is as a broker, so the priority of auction notification events is lower than that of client requests. When the agent receives a request to bid in an auction, it checks that the requesting agent is a client and that the client has sufficient credit before making the bid. When it receives notification of a new auction, the agent may decide to bid on its own account. Determining what price it should offer depends on the type of good offered for sale. We require that scheduled intentions complete by their deadlines with probability $\alpha = 0.9$.

The agent's program is shown below.

```
Beliefs:
  client(client1)
  client(client2)
  credit(client1, price1)
  credit(client2, price2)
  credit(agent, price3)
  buys(good1)
  buys(good2)

Plans:
+!bid(C, A, P,) : client(C) <-
  ?credit(C, P); send-bid(A, P).

+auction(A, G) : buys(G) <-
  price(G, P); ?credit(agent, P); send-bid(A, P).
```

At time 1000 the agent receives the following events.

`+!bid(client1, a101, price1)[1100, 20]` A request from client1 to bid price1 in auction a101. The deadline for this event is 1050 and the client is important so the priority of this event is 20.

`+!bid(client2, a102, price2)[1010, 15]` A request from client2 to bid price2 in auction a102 with deadline 1010 and priority 15.

`+auction(a201, good1)[1060, 10]` A notification of an auction a201 offering good1 with deadline 1060 and priority 10.

These events trigger instances of the two plans in the agent's program to give three candidate intentions, τ_1 (plan 1 with triggering event client1), τ_2 (plan 1 with triggering event client2) and τ_3 (plan3 with triggering event a201). The expected execution time of τ_1 and τ_2 (plan 1) at the specified confidence level $\alpha = 0.9$ is 20. The expected execution time of τ_3 (plan 2), which involves determining what price the agent should bid, is 50. τ_2 is not feasible, and is dropped. τ_1 and τ_3 are feasible and are scheduled in deadline order: τ_3 is scheduled first from 1000–1050, as it has the earliest deadline, followed by τ_1 from 1050–1070. The agent starts execution of τ_3.

Consider a new event arriving at time 1030 while τ_3 is still executing (e.g., after the step `price(good1, P)` has been executed but not `?credit(agent, P)`). The new event is a a request for the agent to bid in auction a103 for client2: `+!bid(client2,a103, price2)`. The deadline is 1065 and priority is 15, and the resulting candidate intention, τ_4 has an expected execution time of 20. τ_1 and τ_4 are inserted into the new schedule in deadline order. However it is not possible to schedule τ_3 by its deadline—its expected completion time exceeds its deadline by 20. (Note that it doesn't matter the order in which τ_3 and τ_4 are scheduled, they cannot both be achieved by their deadline.) τ_3 is therefore dropped, and the agent begins to execute τ_4.

While the plans and events in this example are extremely simple, it illustrates how the agent continually updates its scheduled intentions in response to events to give a priority maximal set of intentions that can be achieved by their deadlines with confidence α.

3. REAL-TIME AGENCY

In this section we show that under certain assumptions (which we believe are reasonable for real-time applications), the time required to execute a single cycle of the AgentSpeak(RT) interpreter (and hence the reactivity delay of the agent) is bounded. We also show that an AgentSpeak(RT) agent commits to a priority-maximal set of intentions, and that, given a fixed schedule, the probability that an intention will complete successfully by its deadline is α.

We make the following assumptions about the agent's program and task environment:

1. the set of possible beliefs has a fixed maximal size (for example, the set of possible beliefs can be restricted to the set of ground instances of any atomic formula appearing in a belief context or a test goal for a finite set of constants);
2. the set of possible goals has a fixed maximal size (for example, the set of possible goals can be limited to the set of ground instances of any atomic formula appearing in an achievement goal for a finite set of constants);
3. the maximal possible interval between the arrival time and deadline of any event is a constant d_{max};
4. the minimal expected execution time for any plan is a constant t_{min}; and
5. there is a maximal expected execution time, t_{max}, for any action in the agent program (i.e., $t_{max} = max(et(a, \alpha))$ for any action a at the specified α)

THEOREM 1. *If the sets of possible beliefs and goals, the maximal expected action execution time and the maximal distance to deadline have a fixed maximal size, and the minimal plan execution time has a fixed minimal size, then the time required to execute a single cycle of the AgentSpeak(RT) interpreter is bounded by a constant δ_c.*

PROOF. The time required to compute evt depends on the size of the sets P and G. If the set of all possible beliefs is limited to a fixed finite set of ground belief atoms (assumption 1 above), then the number of possible percepts $|P|$ is bounded by a constant

(assuming that the agent's percepts are limited to changes in its beliefs).

If the set of all possible agent goals is similarly limited (assumption 2), then the number of possible goals $|G|$ is also bounded by a constant. This means that $|E|$ and $|B|$ are also bounded by a constant at all stages of the cycle.

The time required to compute opt is bounded if $|B|$ is bounded. Computing the set of applicable plans for each event involves evaluating the belief context of each plan whose trigger matches the event against the agent's beliefs. Assuming that returning the set of plans which match an event is a constant time operation and matching the belief context of a plan against the agent's beliefs is bounded by a polynomial in $|B|$, if $|B|$ is bounded, then the time required to compute opt is also bounded by a constant. Computing $sched$ is bounded by a polynomial (in fact, a quadratic function) in $|I|$. In the worst case, when priority varies with deadline and intentions are inserted into the schedule in order of decreasing deadlines, then the feasibility of each new intention involves checking the feasibility of all currently scheduled intentions. $|I|$ is bounded if the maximal possible interval between the arrival time and deadline of any event is a constant d_{max} (assumption 3), and the minimal expected execution time for any plan is a constant t_{min} (assumption 4). Then the maximal possible number of schedulable intentions is bounded by d_{max}/t_{min}. By assumption 5, maximum action execution time and hence the time to compute $exec$ is bounded by a constant t_{max}.

The total cycle execution time is bounded by a constant δ_c which is the sum of the bounds on computing each of the functions. □

By the *reactivity delay* of an agent we mean the time the system takes to recognize and respond to an external event [4] (i.e., the time from the arrival of the event to the selection of a plan for the event or deciding not to respond to the event).

THEOREM 2. *Given assumptions 1-5, the reactivity delay of an AgentSpeak(RT) agent is bounded.*

PROOF. The maximum reactivity delay is for an event which arrives just after the evaluation of evt begins, which is guaranteed to be responded to by the end of the *next* agent cycle. Since the agent's cycle is bounded by δ_c, the maximum reactivity delay is hence bounded by $2\delta_c$. □

Note that the analogous result does not hold for AgentSpeak(L) [9], even when the set of beliefs and goals are bounded. AgentSpeak(L) processes a single event per cycle, and the order in which events are processed is determined by the event selection function $\mathcal{S}_\mathcal{E}$. If $\mathcal{S}_\mathcal{E}$ preferentially returns events of a particular type and events of this type arrive sufficiently frequently, then other events will never be processed.

We now show that an AgentSpeak(RT) agent commits to a priority-maximal set of intentions.

DEFINITION 1. *Consider a set of intentions I. A set $\Gamma \subseteq I$ is a priority-maximal set of intentions (with respect to I) if:*

1. *Γ is feasible;*
2. *$\forall \tau \in I$ such that $\tau \notin \Gamma$: $\{\tau\} \cup \Gamma$ is infeasible;*
3. *$\forall \tau \in I$ such that $\tau \notin \Gamma$, either $\{\tau\}$ is infeasible, or $\exists \Gamma' \subseteq \Gamma$: the minimal priority of an intention in Γ' is greater or equal to $p(\tau)$, and $\Gamma' \cup \{\tau\}$ is infeasible.*

Intuitively, this definition describes a subset of I which is 'maximally feasible' (no more intentions from I can be added if the intentions are to remain feasible at the specified confidence level) and moreover, intentions in $I \setminus \Gamma$ are incompatible with some subset of Γ which contains higher priority intention(s). Observe that if all intentions in I have a unique priority, then there is only one priority-maximal subset of Γ, containing the maximal number of highest priority intentions which are jointly feasible.[6]

THEOREM 3. *Given a partially ordered set of intentions $I = \{\tau_1, \tau_2, \ldots, \tau_n\}$, where $p(\tau_i) \geq p(\tau_j)$ for $i < j$, the AgentSpeak(RT) scheduling algorithm generates a priority-maximal set of intentions $\Gamma \subseteq I$.*

PROOF. Note that the AgentSpeak(RT) scheduling algorithm generates a sequence of sets starting with $\Gamma_0 = \emptyset$, and sets Γ_i to be $\Gamma_{i-1} \cup \{\tau_i\}$, $\tau_i \in I$ if $\Gamma_{i-1} \cup \{\tau_i\}$ is feasible in deadline order, or Γ_{i-1} otherwise. The last set Γ_n is Γ. By construction, Γ is a feasible set of intentions. Γ is also clearly a maximally feasible subset of I: there is no $\tau \in I$ such that $\tau \notin \Gamma$ and $\Gamma \cup \{\tau\}$ is feasible. To prove that it is priority-maximal, let $\tau_i \in I$, $\{\tau_i\}$ feasible, and $\tau_i \notin \Gamma$. We need to show that τ_i is incompatible with some subset of Γ which contains only intentions of the same or higher priority than $p(\tau_i)$. Since the intentions are added to Γ in descending order of priority, when τ_i is considered and found incompatible with Γ_{i-1}, $p(\tau_i) \leq \min(\{p(\tau') : \tau' \in \Gamma_{i-1}\})$. □

THEOREM 4. *The probability that an intention τ will execute successfully in a static environment is equal to α.*

PROOF. Immediate, from the fact that the execution time profiles of plans give us the estimate of duration of the task with the probability α. □

4. DYNAMIC ENVIRONMENTS

The guarantees in the previous section are for a static environment and schedule. They do not consider cases where a new intention generated by the arrival of an event cannot be scheduled, or a scheduled intention is dropped as a result of the arrival of a higher priority task. In this section we develop a simple model of task arrival which can be used to characterise the 'difficulty' of an agent's task environment. We show how this model can be used to determine the priority of intentions which can be reliably scheduled in an environment of specified difficulty, and to estimate the probability that an intention of given priority will not be displaced from the schedule by the arrival of an intention of higher priority.

We characterise the agent's task environment in terms of the average arrival rate and time available for the execution of intentions of a given priority. Let r_i be the average triggering rate of intentions of priority i (expressed as the number of triggering events / unit time), and a_i the average time available for their execution, i.e., the difference between the intention's deadline and the time at which it was triggered. For example, if each external achievement goal has a distinct priority level, r_i and a_i correspond to the arrival rate and average time to achieve a particular type of goal. We assume that $a_i \geq t_i$ where t_i is the average execution time of intentions of priority i at the specified confidence level α, i.e., that deadlines advance with the time at which intentions are triggered

[6]In general, a priority-maximal set of intentions is not guaranteed to contain the largest number of high priority intentions. For example, if $S = \{\tau_1, \tau_2, \tau_3, \tau_4\}$, where $p(\tau_1) = p(\tau_2) = p(\tau_3) = 2$, $p(\tau_4) = 1$, and it is possible to schedule either τ_1 and τ_4 together, or τ_2 and τ_3 together, both sets $\{\tau_1, \tau_4\}$ and $\{\tau_2, \tau_3\}$ will be priority-maximal sets (but, for example, $\{\tau_1\}$ will not be). Computing the set containing the largest number of highest priority intentions is a hard combinatorial problem, which can not be solved by a real-time scheduler.

such that intentions are always individually feasible on average. t_i can be computed from the execution time profiles of the plans in the agent's plan library for the task environment. The larger r_i and the smaller the difference between a_i and t_i, the more difficult the agent's environment. The larger the value of r_i the greater the number of intentions the agent must execute in a given period of time; the smaller the value of $a_i - t_i$ the less time there is to accommodate intentions of priority less than i. In general, the probability that an intention of priority j will be unschedulable is an increasing function of r_i and decreases with $a_i - t_i$ for all $i > j$.

In the worst case, when the schedule is full and intentions complete their execution just before their deadlines, the long term average number of intentions of priority i in the agent's schedule is given by $\lambda_i = r_i a_i$. The amount of uncommitted or 'slack' time unused by intentions of priority i in such a schedule is $s_i = \lambda_i(a_i - t_i)$. We assume that $s_i \geq 0$ for all i given an otherwise empty schedule, i.e., that the average arrival rate and time available for execution of intentions of each priority level are feasible for the agent. For intentions of priority $i-1$ to be reliably scheduled, the total time required for their execution, $\lambda_{i-1}\, t_{i-1}$, must be less than s_i. If the maximum priority of any intention is m, then the time available to schedule intentions of priority j is

$$s_{j+1} = \lambda_m(a_m - t_m) - \sum_{i=j+1}^{m} \lambda_i t_i$$

Hence intentions of priority $j < m$ are typically unschedulable if $s_{j+1} \ll \lambda_j t_j$. For given values of r_i, a_i and t_i, we can therefore determine the priorities of intentions which can be typically scheduled.

For a new intention of priority j to be schedulable, there must be at least t_j slack in the schedule at level j, i.e., $s_j \geq t_j$. The amount of slack at priority level j in the schedule depends on the number of intentions in the schedule at priority levels $j, \ldots, m$. (A new intention of priority j can displace already scheduled intentions with priority $< j$ but not already scheduled intentions of priority j or higher.) Any currently scheduled intentions of priority i, $j \leq i \leq m$, must have arrived in the last a_i time units, i.e., between $-a_i$ and now. The number of intentions of each priority level $j, \ldots, m$ arriving between times $-a_j, \ldots, -a_m$ and now can be represented as a vector $\langle f_j, \ldots, f_m \rangle$ where f_i for $i \in \{j, \ldots, m\}$ is the number of intentions of priority i which arrive within the last $-a_i$. Thus, for an intention of priority j to be schedulable, the following must hold:

$$t_j \leq f_m(a_m - t_m) - \sum_{i=j}^{m} f_i t_i$$

Let the set of vectors satisfying this condition be

$$F \quad = \{\langle f_j, \ldots, f_m \rangle : \quad f_m(a_m - t_m) - \sum_{i=j}^{m} f_i t_i \geq t_j\}$$

The probability that an intention of priority j is schedulable, F_j, is then the probability that at most the number of intentions of each priority level specified by one such sequence of arrivals occurs. If the arrival of triggering events is a Poisson process, the probability that at most f_i intentions are added to the schedule in time a_i is given by

$$F(f_i) = \sum_{x=0}^{f_i} \frac{e^{-\lambda_i} \lambda_i^x}{x!}$$

That is, the probability that exactly 0 or 1 or 2 or ... or f_i intentions are added to the schedule in an interval of length a_i. The probability that at most the number of intentions of each priority level specified by one such sequence occurs is then

$$F_j = 1 - \prod_{\langle f_j, \ldots, f_m \rangle \in F} (1 - F(f_j) \times \ldots \times F(f_m))$$

We can also determine the probability that a scheduled intention of priority j is displaced from the schedule by the arrival of a higher priority intention. If the uncommitted time at priority m, s_m, is sufficient to schedule the expected number of intentions of priority $m-1$, then for an intention of priority $m-1$ to be displaced from the schedule, $u_m = \lceil s_{m-1}/t_m \rceil$ intentions must be added to the schedule during time a_{m-1}. The expected number of priority m intentions arriving in time a_{m-1} is $\lambda_m = r_m a_{m-1}$. The probability that at least u_m intentions are added to the schedule in time a_{m-1} is given by

$$U(u_m) = 1 - F(u_{m-1})$$

That is, 1− the probability that exactly 0 or 1 or 2 or ... or u_{m-1} events arrive in an interval of length a_{m-1}.

In general, for a scheduled intention of priority $j < m$ to be displaced, sufficient intentions of priority $> j$, with total execution time $> s_j$, must arrive within a time interval a_j. A set of intentions with priorities $j+1, j+2 \ldots, m$ sufficient to displace an intention of priority j can be represented as a vector $\langle u_{j+1}, \ldots, u_m \rangle$ where $u_i \in \{j+1, \ldots, m\}$ is the number of intentions of priority i which arrive within a_j. To displace an intention of priority j such vectors must satisfy a number of conditions. First, the number of intentions of each priority must be feasible given s_j. Second, the combined execution time of all intentions in the set must be greater than s_j. Third, that the combined execution time of the intentions should exceed s_j by at most the least execution time of any intention in the set. Let the set of vectors satisfying the conditions be

$$\begin{aligned} U \quad = \{\langle u_{j+1}, \ldots, u_m \rangle : \quad & 0 \leq u_i \leq s_j/t_i, \\ & \textstyle\sum u_i t_i > s_j, \\ & \textstyle\sum u_i t_i - \min_{i \in \{j+1, \ldots, m\}}(t_i) \leq s_j\} \end{aligned}$$

That is, the set of all possible sequences of intentions of priority $> j$ which have combined execution time "just greater" than s_j. The probability that an intention of priority j is displaced, U_j, is then the probability that at least the number of intentions of each priority level specified by one such sequence occurs

$$U_j = 1 - \prod_{\langle u_{j+1}, \ldots, u_m \rangle \in U} 1 - (U(u_{j+1}) \times \ldots \times U(u_m))$$

where $U(u_i) = 1 - \sum_{x=0}^{u_i - 1} \frac{e^{-\lambda_i} \lambda_i^x}{x!}$ as above.

The probability that an AgentSpeak(RT) agent will execute an intention of priority j to completion is then

$$E_j = F_j \times (1 - U_j) \times \alpha$$

For given values of r_i, a_i and t_i, we can therefore determine the probability that an intention of given priority will not be scheduled, or will be displaced from the schedule by a higher priority intention before it can complete successfully. For example, given the rate at which requests to bid arrive and the time it takes to execute the agent's plan to process bids, we can determine the probability that an intention to bid will be executed. For different applications and priority levels, different probabilities of execution may be appropriate. If, for the intended application, E_j is deemed to be too low, the agent developer must either reduce the average triggering rate of intentions of priority $> j$ or increase $a_i - t_i$ for $i \geq j$, e.g., by reducing the execution time of the agent's plans.

5. RELATED WORK

A number of agent architectures and platforms have been proposed for the development of agents which must operate in highly dynamic environments. For example, the Procedural Reasoning System (PRS) [5] and PRS-like systems, e.g., JAM [6] and SPARK [8], have features such as metalevel reasoning which facilitate the development of agents for real time environments. However, to guarantee real time behaviour, these systems have to be programmed for each particular task environment—there are no general methods or tools which allow the agent developer to specify that a particular goal should be achieved by a specified time or that an action should be performed within a particular interval of an event occurring. In contrast, AgentSpeak(RT) provides a high-level programmatic interface to a standardised real-time reasoning mechanism for tasks with different priorities and deadlines.

Perhaps the work most similar to that described here are architectures such as the Soft Real-Time Agent Architecture [12] and AgentSpeak(XL) [1]. These architectures use the TÆMS (Task Analysis, Environment Modelling, and Simulation) framework [3] together with Design-To-Criteria scheduling [13] to schedule intentions. TÆMS provides a high-level framework for specifying the expected quality, cost and duration of of methods (actions) and relationships between tasks (plans). Like AgentSpeak(RT), methods and tasks can have deadlines, and TÆMS assumes the availability of probability distributions over expected execution times (and quality and costs). DTC decides which tasks to perform, how to perform them, and the order in which they should be performed, so as to satisfy hard constraints (e.g., deadlines) and maximise the agent's objective function. In comparison to AgentSpeak(RT), TÆMS allows the specification of more complex interactions between tasks, and DTC can produce schedules which allow interleaved or parallel execution of tasks. However the view of 'real-time' used in these systems is different from that taken by AgentSpeak(RT). Deadlines are not hard (tasks still have value after their deadline) and no attempt is made to offer probabilistic guarantees regarding the successful execution of tasks. In addition, although DTC can be used in an 'anytime' fashion, neither SRTA or AgentSpeak(XL) execute in bounded time. In [11] we described ARTS, a version of PRS which allows the specification of deadlines and priorities, but which does not provide real-time guarantees for the execution of intentions.

6. CONCLUSIONS

The AgentSpeak(RT) architecture provides a flexible framework for the development of real-time BDI agents. An AgentSpeak(RT) agent will achieve a priority-maximal set of intentions by their deadlines with specified confidence. If not all intentions can be achieved by their deadlines, the agent prefers intentions with greater priority. By varying the confidence level, the developer can control the degree of 'optimism' the agent adopts when determining the time required to complete a task in a given environment. In task environments requiring a higher level of confidence, the agent will typically allow more time to complete tasks (and so schedule fewer tasks). As tasks are scheduled in priority order, increasing the level of confidence also has the effect of causing the agent to focus more on high priority tasks at the expense of lower priority tasks which might be achievable given a more optimistic view of execution time. Real-time tasks can be freely mixed with tasks for which no deadline and/or priority is specified. Tasks without deadlines will be processed after any task with a specified deadline. If no deadlines or priorities are specified, the behaviour of the agent defaults to that of a non real-time BDI agent. Although the approach to real-time BDI agents we have presented here has been developed in the context of a particular BDI agent programming language, we believe it can be applied to other BDI-based agent programming languages.

AgentSpeak(RT) adopts a single-threaded execution model. While this is appropriate for the majority of real-time applications where the execution times of intentions are relatively short, there are situations where it would be advantageous to be able to execute long-running actions and plans in parallel with other intentions. In future work we plan to extend the AgentSpeak(RT) architecture to allow the parallel execution of intentions as in [2, 10] and investigate alternative approaches to handling plan failure.

7. REFERENCES

[1] R. Bordini, A. L. C. Bazzan, R. de Oliveira Jannone, D. M. Basso, R. M. Vicari, and V. R. Lesser. AgentSpeak(XL): Efficient intention selection in BDI agents via decision-theoretic task scheduling. In *Proc. of the 1st Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS'02)*, pages 1294–1302, 2002.

[2] B. J. Clement, E. H. Durfee, and A. C. Barrett. Abstract reasoning for planning and coordination. *J. of Artificial Intelligence Research*, 28:453–515, 2007.

[3] K. S. Decker and V. R. Lesser. Quantitative modeling of complex environments. *Int. J. of Intelligent Systems in Accounting, Finance and Management*, 2:215–234, 1993.

[4] M. P. Georgeff and F. F. Ingrand. Decision-making in an embedded reasoning system. In *Proc. of the 11th Int. Joint Conf. on Artificial Intelligence (IJCAI'89)*, pages 972–978.

[5] M. P. Georgeff and A. L. Lansky. Procedural knowledge. *Proc. of the IEEE, Special Issue on Knowledge Representation*, 74(10):1383–1398, 1986.

[6] M. J. Huber. JAM: a BDI-theoretic mobile agent architecture. In *Proc. of the 3rd Annual Conference on Autonomous Agents (AGENTS'99)*, pages 236–243, 1999.

[7] J. Y.-T. Leung and J. Whitehead. On the complexity of fixed-priority scheduling of periodic, real-time tasks. *Performance Evaluation*, 2(4):237–250, 1982.

[8] D. Morley and K. Myers. The SPARK agent framework. In *Proc. of the 3rd Int. Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS'04)*, pages 714–721, 2004.

[9] A. S. Rao. AgentSpeak(L): BDI agents speak out in a logical computable language. In *Proc. of the 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World: Agents Breaking Away*, pages 42–55, 1996.

[10] J. Thangarajah and L. Padgham. An empirical evaluation of reasoning about resource conflicts. *Proc. of the 3rd Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS'03)*, 3:1298–1299, 2004.

[11] K. Vikhorev, N. Alechina, and B. Logan. The ARTS real-time agent architecture. In *Languages, Methodologies, and Development Tools for Multi-Agent Systems*, Vol 6039 of *LNCS*, pages 1–15. 2010.

[12] R. Vincent, B. Horling, V. Lesser, and T. Wagner. Implementing soft real-time agent control. In *Proc. of the 5th Int. Conf. on Autonomous Agents (AGENTS'01)*, pages 355–362, 2001.

[13] T. Wagner, A. Garvey, and V. Lesser. Criteria-directed heuristic task scheduling. *Int. J. of Approximate Reasoning*, 19:91–118, 1998.