On the Life-Cycle of BDI Agent Goals

John Thangarajah [1] and **James Harland** [2] and **David Morley** [3] and **Neil Yorke-Smith** [4]

Introduction. Deliberation over courses of action to pursue is fundamental to agent systems. Agents designed to work in dynamic environments, such as a rescue robot or an online travel agent, must be able to reason about what actions they should take, incorporating deliberation into their execution cycle, reviewing decisions and taking corrective action with appropriate focus and frequency. Not only must agents reason about the effects of their courses of action, they must also consider the semantics of these corrective actions.

Systems based on the well-known *Belief-Desire-Intention* (BDI) framework most often ascribe a set of *goals* to the agent, which is equipped with various techniques to deliberate over and manage this set. The centrality of reasoning over goals is seen in the techniques investigated in the literature, which include subgoaling and plan selection, detection and resolution of conflicts or opportunities for cooperation [9], checking goal properties to specification [10, 5], failure recovery and planning [2], and dropping, aborting, or suspending and resuming goals [7].

A variety of goals are described in the literature, including goals of *performance* of a task, *achievement* of a state, *querying* truth of a statement, *testing* veracity of beliefs, and *maintenance* of a condition [1, 11]. An agent must manage such a variety of goals, while incorporating pertinent sources of information into its decisions over them, such as preferences, quality goals, motivational goals, and advice [10]. The complexity of agent goal management stems from this combination of the variety of goals and the breadth of deliberation considerations. It is furthered because each goal can be dropped, aborted, suspended, or resumed (as illustrated in Figure 1) at arbitrary times. While goals themselves are static (i.e., they are specified at design time, and do not change during execution), their behaviour is dynamic: a goal may undergo a variety of changes of state during its execution cycle [5]. This evolution may include its initial adoption by the agent, being actively pursued, being suspended and then later resumed, and eventually succeeding (or failing). (Maintenance goals have a subtle life-cycle: the goal is retained even when the desired property is true; it is possible that such goals are never dropped.)

Our work analyzes the behaviour of the above types of goals, including the behaviour when goals are aborted or suspended. We consider the complete life-cycle of goals, from their initial adoption by the agent to the time when they are no longer of interest, and all stages in between; we account for the dynamics of plan execution and sub-goaling. We develop a generic framework for goal states and transitions that captures the life-cycle of goals—shown in summary in Figure 1; the Active and Suspended states decomposed further [8]—

[1] RMIT University, Melbourne, Australia, *johnt@rmit.edu.au*
[2] RMIT University, Melbourne, Australia, *james.harland@rmit.edu.au*
[3] SRI International, Menlo Park, USA, *morley@AI.SRI.COM*
[4] SRI International, USA, and American University of Beirut, Lebanon, *nysmith@aub.edu.lb*

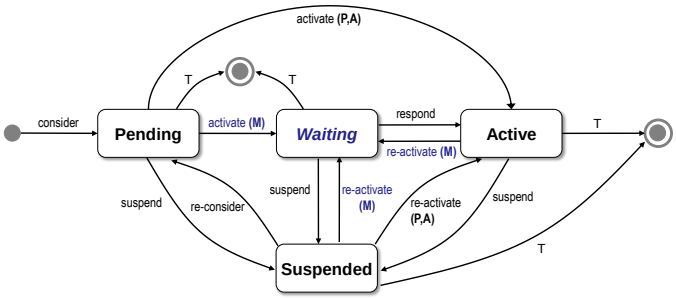

Figure 1. Goal life-cycle composed of abstract states. P – perform goal, A – achieve goal, M – maintain goal, ⊤ – drop/abort/succeed/fail.

and a formal operational semantics that specifies the behaviour. The value of this work for agent designers is a comprehensive and principled set of mechanisms for goal management.

Scenario. Consider a team of three robots—Alpha, Bravo and Charlie—that are searching for the survivors of an air crash. Each has a limited battery life, and must return to its base to recharge within four hours. The three robots search individually for survivors; when one is found, each robot may call on the others for assistance to bring the survivor to the base.

Initially Alpha is told to search a particular area. After 30 minutes, Alpha finds a survivor with a broken leg. Alpha calls for help from Bravo, as it will require at least two robots to carry the survivor. Once Bravo arrives, both robots carry the survivor back to the base, and then both resume searching. A little later, Alpha receives a call for help from Bravo, who has found another survivor. It takes longer than expected for Alpha to get to the location. Before Alpha arrives, another message from Bravo is received, stating that the survivor has been transported back to the base and so Alpha's assistance is no longer required. Alpha resumes its search. Later it receives a call for help from Charlie, who has found a survivor. Once Charlie's survivor is safely back at the base, Alpha considers resuming its search, but as it has only 30 minutes of battery life left, and as it predicts that it will take at least 15 minutes of travel time to get to where it needs to be, Alpha decides to recharge. Once this is done, Alpha resumes its search. Eventually it completes searching its given area, finding no more survivors, and returns to the base.

This example illustrates some of the complexity and richness of goal deliberation and management and the need for a comprehensive and principled approach. In the scenario, Alpha initially adopts the performance goal of searching its assigned area; this goal is suspended when a survivor is found, and later resumed. In the interim times, Alpha adopts achievement goals (getting survivors to the base), which it may have to abort (when Alpha is too late to help

Bravo). Alpha also has the important maintenance goal to monitor its power usage and recharge when appropriate.

Goal Types. We consider three canonical types of goals:

perform(τ, S, F): accomplish a task τ These *goals-to-do* demand that a set of plans be identified to perform a task. A perform goal succeeds if one or more of its plans complete execution; it fails otherwise, such as if no plan is applicable or all applicable plans fail to execute. Hence, the success condition S will express that "one of the plans in the given set succeeds" to accomplish τ [12, 11]. The perform goal also has a failure condition, F. If F is true at any point during execution, the goal terminates with failure, and execution of all plans is terminated. *Example:* Search a particular area for survivors.

achieve(S, F): reach a state S These *goals-to-be* generate plans to achieve a state, S, and should not be dropped until the state is achieved or is found to be unachievable, signified by the condition F. An achieve goal differs from a perform goal in that it checks its success condition during plan execution and after a plan completes. If the success condition S is true (at any point during execution), the goal terminates successfully; if the failure condition F is true (at any point during execution), the goal terminates with failure. Otherwise, the goal returns to plan generation, even if the previous plan completed successfully. *Example:* Ensure a survivor gets to the base.

maintain(C, π, R, P, S, F): keep a condition C true Maintenance goals monitor a *maintenance condition*, C, initiating a *recovery goal* to restore the condition to true when it becomes false. More precisely [3] we allow a maintain goal to be *reactive*, waiting until the maintenance condition is found to be false, $\mathbf{B} \models \neg C$, and then acting to restore it by adopting a reactive recovery goal R; or to be *proactive*, waiting until the condition is predicted to become false, $\mathbf{B} \models \pi(\neg C)$ (where π is some prediction mechanism, say using lookahead reasoning, e.g., [9], and $\mathbf{B}$ is the set of agent's beliefs) and then acting to prevent it by adopting a proactive preventative goal P. Although not specified in prior work, we insist that R and P be achieve goals. The maintenance goal continues until either the success condition S or failure condition F become true. *Example:* Ensure that Alpha is always adequately charged.

Contribution. We have developed a rich and detailed specification of the appropriate operational behaviour when a goal is pursued, succeeded or failed, aborted, suspended, or resumed. We **(1)** include sophisticated maintenance goals, along the lines of Duff et al. [3], that encompass proactive behaviour (i.e., anticipating failure of a given condition) as well as reactive behaviour (i.e., waiting until the condition becomes false), and allow for different responses in each case. This contrasts over most work on maintenance goals, in which only the reactive behaviour is developed [11, 5]. We **(2)** develop an appropriate set of states for goals (which generalizes the two states of suspended and active of van Riemsdijk et al. [11]), and a set of operations to move goals between these states. These operations are richer than previous works, by including suspending and resuming for all the common goal types, and the corresponding state transitions can be non-trivial. We provide a detailed specification elsewhere [8].

Our second area of innovation is to address execution of plans to achieve goals within our semantics. The spirit of our work is shared by Morandini et al. [5], who build on van Riemsdijk et al. by providing operational semantics for non-leaf goals, i.e., semantics for subgoaling and goal achievement conditions. We **(3)** encompass the same dynamic execution behaviour, but further consider plans as well as goals. Thus we consider the execution cycle, not only the design phase like Morandini et al.

We have designed a complete set of operational semantics using the CAN agent specification language [6] and developed a prototype implementation. The scenario described in this paper is successfully executed. By developing the formal operational semantics for our generic framework in CAN, we have not been tied to any particular agent implementation. The prototype implementation, denoted *Orpheus*, consists of around 700 lines of platform-netural Prolog. It is available from the authors at `http://www.cs.rmit.edu.au/~jah/orpheus`.

Summarized, the three key contributions of our generic framework for goal states and transitions are (1) to encompass both goals of accomplishment and rich goals of monitoring, (2) to provide the first specification of abort and suspend for all the common goal types, and (3) to account for plan execution as well as the dynamics of subgoaling. To the best of our knowledge, no existing framework for goal operation accounts for all of these points.

Our work so far accounts for the life-cycle of each goal. We have not sought to address overall agent deliberation, plan deliberation, resource management, or plan scheduling. Thus far we have examined the same questions as Braubach et al. [1]; future research is to address the other questions they pose. Likewise, we have not considered failure handling and exceptions. Our work is complementary to generic or application-specific reasoning about goal interactions (e.g., [9]), and to goal and plan selection. Future work is to establish and prove properties of the semantics (compare the top-down semantic approach of Khan and Lespérance [4]), and to explore its use within agent programming languages.

REFERENCES

[1] L. Braubach, A. Pokahr, D. Moldt, and W. Lamersdorf, 'Goal representation for BDI agent systems', in *Proc. of ProMAS'04*, (2004).

[2] L. de Silva, S. Sardina, and L. Padgham, 'First principles planning in BDI systems', in *Proc. of AAMAS'09*, (2009).

[3] S. Duff, J. Harland, and J. Thangarajah, 'On proactivity and maintenance goals', in *Proc. of AAMAS'06*, (2006).

[4] S. M. Khan and Y. Lespérance, 'A logical framework for prioritized goal change', in *Proc. of AAMAS'10*, (2010).

[5] M. Morandini, L. Penserini, and A. Perini, 'Operational semantics of goal models in adaptive agents', in *Proc. of AAMAS'09*, (2009).

[6] S. Sardiña and L. Padgham, 'Goals in the context of BDI plan failure and planning', in *Proc. of AAMAS'07*, (2007).

[7] J. Thangarajah, J. Harland, D. Morley, and N. Yorke-Smith, 'Suspending and resuming tasks in BDI agents', in *Proc. of AAMAS'08*, (2008).

[8] J. Thangarajah, J. Harland, D. Morley, and N. Yorke-Smith, 'Operational behaviour for executing, suspending and aborting goals in BDI agent systems', in *Proc. of 8th Intl. Workshop on Declarative Agent Languages and Technologies (DALT'10)*, (2010).

[9] J. Thangarajah, L. Padgham, and M. Winikoff, 'Detecting and exploiting positive goal interaction in intelligent agents', in *Proc. of AAMAS'03*, (2003).

[10] A. van Lamsweerde, 'Goal-oriented requirements engineering: A guided tour', in *Proc. of Intl. Joint Conf. on Requirements Engineering (RE'01)*, (2001).

[11] M. B. van Riemsdijk, M. Dastani, and M. Winikoff, 'Goals in agent systems: A unifying framework', in *Proc. of AAMAS'08*, (2008).

[12] M. Winikoff, L. Padgham, J. Harland, and J. Thangarajah, 'Declarative and procedural goals in intelligent agent systems', in *Proc. of KR'02*, (2002).