# Estimating the Progress of Maintenance Goals (Extended Abstract)

John Thangarajah
Computer Science & IT
RMIT University, Australia
johnt@rmit.edu.au

James Harland
Computer Science & IT
RMIT University, Australia
james.harland@rmit.edu.au

Neil Yorke-Smith
Olayan School of Business
American University of Beirut
nysmith@aub.edu.lb

## ABSTRACT

We extend our earlier work on quantifying the level of completeness of achievement goals in BDI agents [8], to encompass maintenance goals. We both characterize what it means for a maintenance goal to be partially complete in terms of its relevancy, and sketch an efficient computational mechanism for an agent to compute dynamic estimates of the progress of its maintenance goals. We also discuss the relationship between our computation of progress estimate with an earlier theoretical perspective on BDI goal completeness.

## Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Intelligent agents.*

## General Terms

Design, Theory

## Keywords

Reasoning in agent-based systems; Maintenance Goals; Belief-Desire-Intention theories and models; Goal completeness

## 1. INTRODUCTION

*Maintenance goals* [1, 3, 2, 4] are recognized as an important concept in agent systems based on the Belief-Desire-Intention (BDI) model [5]. A maintenance goal has a particular state of the world that the agent seeks to maintain, i.e., the state must be true, and kept this way indefinitely. For example, an autonomous Mars rover should ensure that wherever it travels, it always maintains sufficient battery change for the journey back to its base. It can treat a maintenance goal reactively (by reattaining the *maintain condition* if it becomes false) or proactively (by taking actions to prevent the condition from becoming false). Since maintenance goals are not dropped once any violation has been restored, they must be treated differently to *achievement goals*, which are usually dropped once the stated condition has been achieved [6].

The importance of reasoning about partially-complete goals was recognized by van Riemsdijk and Yorke-Smith ('vRYS') [9], who developed a theoretical framework in the case of achievement goals. Subsequently, in a separate line of work, Thangarajah et al. ('Tetal') [8] looked at the practical issue of computing partial completeness estimates for an achievement goal. Neither considered maintenance goals.

By its nature, a maintenance goal is intended to persist in order to uphold its maintain condition—until the goal is no longer relevant. In this sense, a maintenance goal is never 'complete' in the way an achievement goal can be. We introduce two notions of 'completeness' for maintenance goals in BDI-style agents. First, *permanently complete* (PC) refers to a goal no longer being relevant, and hence the agent no longer needing to uphold the goal's maintain condition. By contrast, *recovery complete* (RC) refers to the agent's progress in restoring the maintain condition after it has been violated. To our knowledge, this work is the first study of the concept of progress of maintenance goals in BDI agents.

**Appears in:** *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), Bordini, Elkind, Weiss, Yolum (eds.), May 4–8, 2015, Istanbul, Turkey.*


## 2. SCENARIO AND MECHANISM

We illustrate the role of maintenance goals by extending the Mars rover scenario as given by Thangarajah et al. [8] to include maintenance goals. This autonomous vehicle has been given the task of exploring a particular region ($red1$), which involves investigating a particular rock $rock1$ and moving to a canyon, which it will survey.

The rover's goals include the maintenance goal $M_1$, which must maintain the battery charge at 30% or more. To do this, the rover may adopt either the the recovery goal PauseAndRecharge—the rover stops movement and science and waits until solar cells have recharged battery sufficiently—or the preventative goal ConserveEnergy—for which the rover might move more slowly, or deprioritize some science targets in order to save power. $M_1$ also contains the *success condition* $S(M_1)$ that explored($rock1$) $\wedge$ explored($canyon$), which means that $M_1$ will be considered irrelevant (and hence complete) if both of these conditions are true.

At the start of mission execution, $M_1$ is 100% relevant, i.e., 0% PC. Part-way through execution of exploration of $rock1$, the maintain condition is violated, and so the rover therefore adopts the achievement goal PauseAndRecharge to restore it. $M_1$ is 0% RC at this point. Once PauseAndRecharge succeeds, $M_1$ is 100% RC again.

Later, once $rock1$ has been explored, the first part of $M_1$'s success condition is true, and the rover computes that $M_1$ is now 50% relevant, i.e., 50% PC. This is clearly an estimate, since there could be more science targets in explored($canyon$) than in explored($rock1$), or it could take more effort to study one target in the latter than in the former. Given that $M_1$ is

still 50% relevant, the rover estimates there are 50% of targets in $red1$ remaining, and it notes that it already has had to recharge once. Hence, the rover may decide to choose to travel at lower speed than normal, because it predicts that will soon have to pause and charge again.

During exploration of the canyon, the rover finds four priority science targets, and so the success condition for $M_1$ is expanded to explored($rock1$) $\wedge$ explored($target1$) $\wedge$ ... $\wedge$ explored($target4$). Note that such refinement of terms in the success condition means that the relevance (PC) estimates of $M_1$ are non-monotonic. Before any of the canyon targets are investigated, the relevance of $M_1$ has fallen from 50% to 20% (i.e., $rock1$ done but four canyon targets remain). As the rover proceeds, $M_1$'s progress estimate updates further.

A further extension to this scenario is to consider the resources required to perform the battery recharging process. Suppose that a certain amount of memory is required by the recharge process, and so it is necessary that a certain amount of memory be available whenever $M_1$ is violated. One possibility would be to ensure that the first step of the recovery goal PauseAndRecharge includes pausing all other goals, in order to ensure that the memory is available. Another possibility, which enables more rational and proactive resource management, is to introduce a second maintenance goal $M_2$, whose task is to ensure that sufficient memory is available. We hence observe that the methods used for resource-based estimation of the completeness of achievement goals can also be used to predict potential violations of resource-related maintenance goals.

To compute these dynamic progress estimates for a maintenance goal $M$, we leverage the mechanisms of Tetal [8]. First, the percentage PC of $M$ at time $t$ is estimated based on estimating the percentage of effects in $S(M)$ that are true at time $t$. Second, when $M$ is not (100%) PC, then it is not (100%) RC, and hence there is a recovery goal $R$ or preventative goal $P$. Since $R$ and $P$ are necessarily achievement goals [6], we can leverage the computation mechanism for completeness for $R$ or $P$, either estimating bounds on the percentage RC of $M$ based on lower- and upper-bound estimates of resource consumption of $R/P$, or based on lower- and upper-bound estimates of the effects of $R/P$.

We have implemented these mechanisms and the above scenario in the abstract agent language CAN [7], and have used it to experiment with the interaction between resource estimates and maintenance goals. The implementation consists of around 2,000 lines of Prolog, and is available from `goanna.cs.rmit.edu.au/~jah/orpheus`.

## 3. THEORETICAL PERSPECTIVE

We now reconcile the computational approach of Tetal with the theoretical framework of vRYS. Our motivation is to ask whether the former can provide the computational mechanism missing in the latter. The key question is what defines (full) completeness of a goal $g$. vRYS suppose each goal has a progress metric, denoted as a set $A$ with a partial order $\leq$ (usually total w.r.t. $a_{\min}$), and further a minimum value, $a_{\min} \in A$, the *completion value*, that should be reached in order to consider the goal to have been completely satisfied. By contrast, Tetal hold the classical view that completeness of $g$ is defined by its success condition.

Since vRYS do not use the logical conjunction of effects in $S(g)$ to define completeness, they require each goal to have a *progress appraisal function* $\psi$ from $\mathcal{S}$, the set of states, to $A$. In addition, they posit an accompanying upper bound function, which we denote $\psi_U$, that takes into account the means $m$ that "will be used for pursuing the goal": $\psi_U$ "yields (an estimation of) the maximum value in $A$ reachable" from a state $s \in \mathcal{S}$" with means $m$. They also mention but do not develop a lower bound function which we denote $\psi_L$. vRYS recognize that all three functions may be estimates, which concurs with the principles of Tetal's mechanism.

Tetal explicitly compute lower- and upper-bounds, an approach we have followed in this work. These bounds can be seen as equivalent to vRYS's $\psi$ (or $\psi_L$) and $\psi_U$ for the progress metric. While vRYS consider only a single such "metric", Tetal compute multiple metrics—multiple resources and multiple effects. However, vRYS note "Besides the metric chosen as the progress metric, the agent (or designer) might have interest in others: e.g., progress may be defined in terms of tracks searched, but time taken could be an additional relevant factor in the team's decisions" [9].

We conclude that the general philosophy of the two lines of work seem compatible. The computation mechanisms of Tetal of lower- and upper-bounds can be seen as computing the $\psi_L$ and $\psi_U$ bounds of the progress metric of vRYS. That is, if one metric of those computed by Tetal is designated as the progress metrics for a goal in vRYS's framework, then Tetal provide the computational mechanism that is lacking in vRYS's framework.

**Acknowledgements.** We thank the reviewers for their comments. JT acknowledges the ARC Discovery grant number DP1094627. NYS thanks the Operations group at the Cambridge Judge Business School and the fellowship at St Edmund's College, Cambridge.

## REFERENCES

[1] L. Braubach, A. Pokahr, D. Moldt, and W. Lamersdorf. Goal representation for BDI agent systems. In *Proc. ProMAS'04*, pages 44–65, 2004.

[2] M. Dastani, M. B. van Riemsdijk, and J.-J. C. Meyer. Goal types in agent programming. In *Proc. ECAI'06*, pages 220–224, 2006.

[3] S. Duff, J. Harland, and J. Thangarajah. On proactivity and maintenance goals. In *Proc. AAMAS'06*, pages 1033–1040, 2006.

[4] S. Duff, J. Thangarajah, and J. Harland. Maintenance goals in intelligent agents. *Computational Intelligence*, 30(1):71–114, 2014.

[5] M. Georgeff and A. Rao. Rational software agents: From theory to practice. In *Agent Technology*, pages 139–160. Springer, New York, 1998.

[6] J. Harland, D. N. Morley, J. Thangarajah, and N. Yorke-Smith. An operational semantics for the goal life-cycle in BDI agents. *JAAMAS*, 28(4):682–719, 2014.

[7] S. Sardiña and L. Padgham. A BDI agent programming language with failure handling, declarative goals, and planning. *JAAMAS*, 23(1):18–70, 2011.

[8] J. Thangarajah, J. Harland, D. N. Morley, and N. Yorke-Smith. Quantifying the completeness of goals in BDI agents. In *Proc. ECAI'14*, pages 879–884, 2014.

[9] M. B. van Riemsdijk and N. Yorke-Smith. Towards reasoning with partial goal satisfaction in intelligent agents. In *Proc. ProMAS'10*, pages 41–59, 2010.