ECAI 2014
T. Schaub et al. (Eds.)

doi:10.3233/978-1-61499-419-0-879

Quantifying the Completeness of Goals in BDI Agent Systems

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

Abstract. Given the current set of intentions an autonomous agent may have, intention selection is the agent's decision which intention it should focus on next. Often, in the presence of conflicts, the agent has to choose between multiple intentions. One factor that may play a role in this deliberation is the level of completeness of the intentions. To that end, this paper provides pragmatic but principled mechanisms for quantifying the level of completeness of goals in a BDI-style agent. Our approach leverages previous work on resource and effects summarization but we go beyond by accommodating both dynamic resource summaries and goal effects, while also allowing a non-binary quantification of goal completeness. We demonstrate the computational approach on an autonomous robot case study.

1 Introduction

In agent systems in the Belief-Desire-Intention (BDI) tradition, the most common conceptualization of goal accomplishment is discrete: a goal is either complete (usually, a plan for it has succeeded), or it is incomplete (whether execution of a plan or plans for it has begun or not) [4, 23]. For the deliberation that an intelligent agent undertakes about its goals – such as the decision about which intention to focus on next – the agent is thus limited to a coarse binary approximation of goal completeness. If the agent were able to compute a finer-grained approximation of the level of completeness of its goals, it could make more nuanced and potentially more suitable decisions. For example, when resolving goal conflicts [20], the agent may choose to continue with the goal that is more complete than the other.

While the notion of partially-complete goals has been defined in the literature [26, 23], reasoning frameworks to date have largely left unanswered *how* to compute the level of completeness of a goal in a realistic and principled manner.

Our focus in this work is *to provide a principled and general approach that can be used computationally to quantify a measure of completeness for a given goal*. It is not our aim here to specify how an agent subsequently uses this information, i.e., its (intention) deliberation mechanisms.

There are a number of factors that may contribute towards assessing the completeness of a goal: resources, deadlines, number of actions/plans complete, time elapsed, effects realized, etc. In this paper, we propose the use of two factors to determine a quantifiable measure of completeness of a goal: *resource consumption* and the *effects* of achieving the goal.

[1] RMIT University, Australia, email: johnt@rmit.edu.au
[2] RMIT University, Australia, email: james.harland@rmit.edu.au
[3] Lingonautic, Inc., email: davidmorley@luckmor.com
[4] American University of Beirut, Lebanon, and University of Cambridge, UK, email: nysmith@aub.edu.lb

First, we use resource consumption to provide a measure of the level of *effort* the agent has dedicated towards satisfying the goal. There has been previous work on representing resource requirements and continuously refining them as the agent executes its goals [22, 14]. We build on this existing work to provide a quantifiable measure of completeness with respect to effort.

Second, the *effects* of a goal capture its desired outcome, generally in terms of conditions that should be true when the goal execution is complete [16, 20]. For example, the effect of a goal of a Mars rover robot to scan an area for targets of interest is that the area is scanned. We use the effects of the goal to provide a measure of the level of goal *accomplishment*, since the purpose of the goal is indeed to bring about its intended effects. As with resources, we build on and extend existing work on representing and reasoning about the effects of goals and plans [21]. In that prior work, effects are represented as boolean predicates, such as *area-scanned* in the rover example. However, there may be instances where the conditions may be satisfied to a certain degree, such as the area is 80% scanned. We extend the prior work to allow for this representation.

Besides these two factors representing effort and accomplishment, we mention two other factors that might seem amenable to be used as a measure of completeness: the number of actions performed by the agent and the time taken. To reason with the number of actions would require the assumptions that all actions are explicitly represented and that the distribution of effort to perform each goal is known. Our experience finds this to be not the case with most practical agent systems developed in languages such as JACK [24], SPARK [13], GORITE [15], etc., where actions are arbitrary code and, crucially, are not explicitly represented. Time, on the other hand, can be measured with respect to the pace of goal execution. However, to reason about the time required to execute a particular goal, an explicit representation of the time taken to execute each action or an entire plan is needed. If this is the case, then it is possible to consider time as a type of resource and use the same computational mechanisms we describe for resources, as we will illustrate. To ensure tractable computation, however, we do not consider dedicated temporal reasoning or projections (compare [23]).

The contribution of this paper, then, is a principled mechanism for computing completeness of top-level goals of a BDI-style agent in order to inform the agent's deliberation. To our knowledge, this work is the first to study such computation with an emphasis on tractable, pragmatic reasoning.[5]

Sect. 2 situates our work in the literature. Sect. 3 describes an autonomous robot case study. Sect. 4 presents our computational approach and its implementation. Sect. 5 suggests future directions.

[5] An abstract sketch of the ideas presented in this paper, without details of the mechanisms, appeared as an extended abstract at AAMAS'14 [18].

2 Background

While goals in agent programming languages are not customarily defined to allow for partial completeness, Holton, from a philosophical stance, argues for the existence of "partial intentions", a concept spanning both desires and goals [8]. Haddawy and Hanks made an early study [6], in which a function from propositions to a real number represents the degree of satisfaction of a goal.

Goals have commonly been associated with a utility, priority, or preference in the literature of agents (e.g., [9, 7, 11]) and of AI planning (e.g., [2]). The purpose is usually for a form of intention selection: which goals to prioritize/pursue, or which plan/action to select.

Thangarajah et al. [19] explore multiple criteria that an agent may include in its goal deliberation, including utility, preference, deadline, resource considerations, goal interactions, effort to date, and likelihood of success. Although they describe a dynamic constraint-based reasoning mechanism, these authors also do not explicitly consider reasoning with partially-complete goals.

Based on their earlier work [22, 21], Thangarajah and Padgham [20] study goal interactions, both positive (synergy) and negative (conflicts). Their work considers action effects as simple boolean predicates. They define the *Goal-Plan Tree* (GPT) structure of alternating layers of goal and plan nodes, and use this structure to inform deliberation such as goal adoption and plan selection. The reasoning centres around the use of resource and effect summaries annotated on GPT nodes and dynamically updated as execution proceeds.

Morley et al. [14] further develop reasoning in a BDI-style agent over GPT structures. They provide an algorithm for an agent to dynamically update resource estimates on GPTs – i.e., as a goal is executed – accommodating resource bound information, parameterized goals, and rich plan constructs. Again unlike our work, they do not explicitly consider reasoning with measures of goal completeness.

Zhou and Chen adopt instead a logical approach, defining a semantics for partial implication of desirable propositions [25]. Zhou et al. [26] investigate partial goal satisfaction on the basis of this logical semantics, viewing a goal as completed when a (possibly disjunctive) proposition is achieved according to the logic. They focus on application of different notions of partial implication to goal modification in the context of belief change.

van Riemsdijk and Yorke-Smith formalize the concept of a partially-complete goal for a BDI-like agent [23]. They capture partial satisfaction of a goal using a progress metric, and a minimum value that the goal must attain for the agent to consider it completely satisfied. They describe agent reasoning using such a representation, but do not provide any detailed computational mechanisms.

As van Riemsdijk and Yorke-Smith point out, there is a body of work on reasoning with partial *plans*, for instance in plan formation or negotiation (e.g., [12, 5, 10]), as well as in the AI planning literature (e.g., [17]). For example, in the area of multi-agent planning and negotiation, Kamar et al. investigate helpful assistance of teammates in pursuit of a plan that could be partially complete [10].

In the context of Hierarchical Task Network (HTN) planning, Clement et al. [1], based on their earlier work, summarize propositional and metric resource conditions and effects (of which [20] can be seen as a special case) of a partial temporal HTN plan, and, like [21], use these to determine potential and definite interactions between abstract tasks. Their work admits resource bound information and emphasizes facilitating the HTN planning process. Although accommodating interleaved local planning, multiagent coordination, and concurrent execution, their work is not in the context of BDI-style agents and does not target measures of goal completeness.

3 Scenario

We illustrate our approach on a Mars rover scenario. An autonomous rover has these resources: spectroscope utilizations, internal memory capacity (for images), and time. The spectroscope involves drilling a small sample from a target (e.g., a rock), and the rover's drill bit has a limited lifetime; hence the spectroscope is a consumable, discrete resource. Memory is a replaceable, discrete resource; and time (seconds) is a renewable but perishable continuous resource. The rover's top-level goal is ExploreRegion($red1$), where $red1$ is a designated region. The rover believes that region $red1$ has a rock of interest, $rock1$, which it is currently near, and an area of interest, $canyon$, within region $red1$ but some distance away from $rock1$, which it has been instructed to survey.

The GPT is shown in Fig. 1. Following the plan *traverseAndStudy* for ExploreRegion($red1$), the rover will perform an Experiment($rock1$) on $rock1$, Traverse($rock1$, $canyon$) (i.e., move) from $rock1$ to $canyon$, and then Survey(canyon).

For an Experiment goal, the rover can choose from two possible plans, one using its spectroscope and the other its thermal image device. In both cases, the rover moves close to the target object, positions its device arm, and performs the measurement and saves any data. For the Survey goal, the rover has a single plan, which is to first IdentifyTargets, which may be fulfilled by a plan that uses the panoramic camera and then selects a target, and second Experiment on the selected target object. (A more elaborate scenario would involve iterating through a list of targets.)

Fig. 1 also shows the effects and the resources estimated to be required for each leaf plan node. We assume these estimations are specified by the rover's designer, e.g., based on past experiences. The resource annotations on plans are single values (when the resource usage can be estimated precisely by the designer), or ranges shown in square brackets (when they cannot be estimated precisely). For simplicity, we work with lower and upper bound range estimates (compare [14, 1]). For example, (time: [20,30]) denotes the required amount of resource time for the plan is estimated to be at least 20 and at most 30 seconds. The remaining annotations are computed from the leaf nodes and goals as we will describe. In particular, note that each goal has a set of success conditions annotated to it, in additional to the effects. More details on this follow in the next section. Note for space reasons we omit the repeated subtree rooted at Experiment($target$) in the right-hand branch of the GPT, as it is indentical to the subtree rooted at Experiment($rock$) apart from the argument.

In order to state an appropriate success condition for goals such as Experiment($rock1$), we use a predicate Measured(X), which is true if either SpectralProfile(X) or ThermalProfile(X) is true. This is easily implemented by an appropriate rule in the agent's beliefs.

4 Quantifying Completeness

Our aim is to provide a principled and general computational approach to quantify a measure of completeness for a given goal. We focus on resource consumption and the effects of achieving the goal.

4.1 Preliminaries

A typical BDI agent system consists of a plan library in which, for a given goal, there are one or more plans that could possibly achieve the goal (OR decomposition); each plan performs some actions or posts a number of subgoals all of which must be achieved (AND decomposition). A subgoal is in turn handled by some plan in the

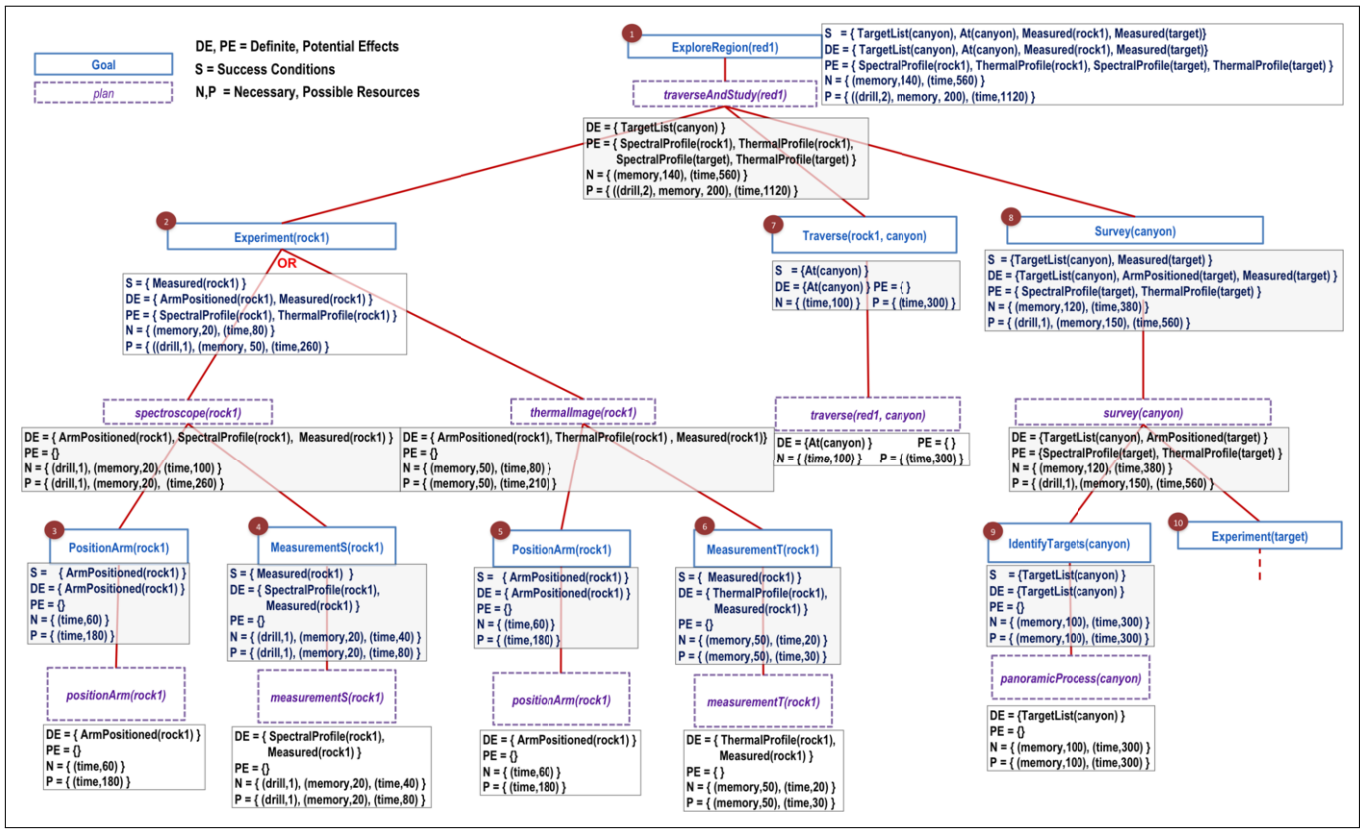

Figure 1. Goal-Plan Tree in the Mars rover scenario. Resources and effects are shown annotated on nodes.

plan library. This decomposition leads to a Goal-Plan Tree structure of the kind illustrated in Fig. 1.

The spirit of our approach follows that of Clement et al. [1] and particularly Thangarajah et al. [20, 21]: we require the goals and plans be annotated with certain information about the resource requirements and effects attained, we generate a Goal-Plan Tree structure with annotations, and use it as described in the sequel.

4.1.1 Resources

The resources consumed when a goal is executed by an agent depends on the plans that are used to achieve the goal. As such, resource requirements are not annotated on goals, but only at the plan level. Following [20, 14], each plan will have ascribed the resources necessary for the plan to complete execution. These do not include the resources required for executing the subgoals of the plan, if any. This declarative specification can made by the agent designer, or in some domains learned from past execution traces.

Def. 1 *A set of resources $\mathcal{R}$ is a set of key-value pairs $\{(r_1,\alpha_1), \ldots, (r_n,\alpha_n)\}$ where r_i is the unique* resource name *and $\alpha_i \in \mathbb{N}$ is its corresponding value.*

For example, in Fig. 1, plan *measurementT* has lower-bound resources {(memory, 50), (time, 20)}. N and P are the necessary and possible resource summaries, defined below in Def. 3.

We consider resources of two types: *consumable* and *reusable*. The former are those that are no longer available following use (e.g., drill bits) and the latter are those that can be reused following usage (e.g., memory, although note in the timescale of the scenario, memory is not reused). In this work we consider both types to be of equal importance, since their relative importance depends on the domain.

4.1.2 Effects

A goal g will have a *success condition*, which describes the state of the world that must be true in order for the goal to be accomplished [16]. We define the success condition to be a set of effects, $S(g)$, where the conjunction of the effects must hold for the goal to be complete (we do not support disjunctions). A plan p will have attached to it the effects attained by the direct actions of that plan [21] excluding the effects of any subgoals that are executed by other plans.

Previous work about effects reasoning [21, 20] defined the effects of a goal (or plan) as simple predicates that are either true or false. As we have seen, this neglects effects which are not discrete but fulfilled continuously to a certain degree. For example, a goal like IdentifyTargets in Fig. 1 involves scanning the canyon for targets, and so may be considered 60% complete once it has scanned 60% of the area of the canyon, which would result in an effect of (*area-scanned*, 60). Let $\mathcal{E}$ be the set of all effect-types relevant to the agent system.

Def. 2 *An* effect *is a key-value pair (e, α) where $e \in \mathcal{E}$, the* effect-type*, is an unique identifying label and $\alpha \in \mathbb{R}$ is the degree to which the effect has to be attained for it to be achieved. For discrete effects, $\alpha \in \{0, 100\}$; for continuous effects, $0 \leq \alpha \leq 100$.*

For example, consider a variant of the goal IdentifyTargets which scans an area and selects a target. Its effects can be represented as: { (*area-scanned*, 80), (*target-selected*, 100) }.

4.1.3 Summary Information

In order to determine the level of completion of a goal g at the current time t, with respect to resources or effects, it is necessary to determine (1) the resources consumed and effects attained thus far in executing g, and (2) the resources required and effects that should be attained in order for the goal to complete from t.

While the former step can be computed accurately, by monitoring the resource consumption and checking the current state of the world for effects achieved, the latter step is more complex. The nature of BDI agent systems are such that there are different ways (plans) of accomplishing a particular goal, and these may use different resources and bring about different effects. Moreover, plans may fail and unexpected events may occur. The deliberation on which way to achieve the goal (i.e., plan selection) is made dynamically during execution depending on the context the agent is in, and hence is not known in advance. Consequently we cannot always say a priori precisely what resources will be needed to accomplish a given goal.

Further, although no matter which way a goal is pursued, its effects ought to be attained, the way in which the goal is achieved may result in further effects. Some of these (side-)effects may be necessary no matter which way the goal is achieved; others not.

Note that the goal Experiment($rock1$) results in the effect *ArmPositioned*($rock1$) no matter which plan is followed, while the effects *SpectralProfile*($rock1$) and *ThermalProfile*($rock1$) depend on the choice of plan used.

The second step above therefore requires some form of look-ahead for both resources and effects. It suffices for us to adopt and extend the efficient look-ahead mechanism of [20, 21] which uses summary information to compute a lower- and upper-bound of future resource usage and effects attained.

Resource Summaries. Previous work used the notion of summary information to estimate the *necessary* (lower-bound) and *possible* (upper-bound) resource requirements of a goal [1, 20]. Necessary resources are those that are used no matter which way the agent chooses to achieve the goal, while possible resources are those that may be needed in the worst case.

In this work, we adopt the algorithms for computing and updating resource summaries as described in [20, 14]. We do not detail the same algorithms here since it is not the contribution of this work and is not necessary to understand the approach we present.

Def. 3 *The dynamically updated* resource summary *of a goal* g *at time* t *is:*

$$RS^t(g) = \langle N_R^t(g), P_R^t(g) \rangle \tag{1}$$

where $N_R^t(g)$ *is the set of* necessary resources *and* $P_R^t(g)$ *the set of* possible resources *required to execute the goal from current time* t.

Effect Summaries. Effect summaries of a goal are defined in terms of *definite* and *potential* effects: definite effects are those that are brought about no matter which way the goal is achieved, while potential effects are those that may potentially be brought about depending on the way the goal is achieved.

Thangarajah et al., similar to their work on resource summaries, presented a set of algorithms for deriving effect summaries at compile time and updating them dynamically at run-time [21].

Def. 4 *The* effects summary *of a goal* g *is:*

$$ES(g) = \langle D_E(g), P_E(g) \rangle \tag{2}$$

where $D_E(g)$ *is the set of* definite effects *and* $P_E(g)$ *is the set of* potential effects *that will be bought about by pursuing the goal* g *at the current time* t.

For example, the goal Survey($canyon$) in Fig. 1 has definite effects D_E(Survey($canyon$)) = {TargetList(canyon), ArmPositioned(target), Measured(target)}, and potential effects P_E(Survey($canyon$)) = {SpectralProfile(target), ThermalProfile(target)}. Note that $D_E(g)$ and $P_E(g)$ are exclusive. Note also that the success condition of the goal is a subset of the definite effects, i.e., $S(g) \subseteq D_E(g)$.

4.2 Resources as a Measure of Completeness

The aim of our resource analysis is to provide an agent with a quantified measure of effort with respect to the amount of resources consumed thus far in executing a goal, in the context of the total resource requirements for achieving the goal. Hence we require the agent to keep track of the total resources consumed in executing each goal.

Def. 5 *Let* $R^t(g)$ *be the set of resources consumed thus far up to current time* t *solely by the execution of* g.

We write $\left(R^t(g)\right)(r)$ for the value of resource r in $R^t(g)$ at time t, i.e., the value α_r (see Def. 1).

Lower-Bound Resource Consumption Analysis. We use the necessary and possible resource summaries to provide a lower- and upper-bound resource consumption analysis, respectively.

Def. 6 *The* lower-bound resource consumption analysis *of a goal* g *at the current time* t *is:*

$$CR_{lb}^t(g) = \frac{\left(\sum_{r \in dom\left(R^t(g) \cup N_R^t(g)\right)} \frac{\left(R^t(g)\right)(r)}{\left(R^t(g) \oplus N_R^t(g)\right)(r)} \right)}{\mid R^t(g) \oplus N_R^t(g) \mid} \tag{3}$$

where dom denotes the domain of the resource types set, i.e., the set of key values (Def. 1). The resource set addition operator ($\oplus$) is defined as: $\mathcal{R}_1 \oplus \mathcal{R}_2 = \{(r, \mathcal{R}_1(r) + \mathcal{R}_2(r)) \mid r \in (dom(\mathcal{R}_1) \cup dom(\mathcal{R}_2))\}$ where r is a resource type and $\mathcal{R}_i(r)$ provides the value of r in the relational set $\mathcal{R}_i \subseteq \mathcal{R}$.

Note that the plausible intuition that CR_{lb}^t is non-decreasing does not hold in general, as in fact can be seen in execution traces in the Mars rover scenario.

Upper-Bound Resource Consumption Analysis. The computation is the same as for the lower bound, except instead of the necessary resource summary we use the *possible* resource summary.

Def. 7 *The* upper-bound resource consumption analysis *of a goal* g *at the current time* t *is:*

$$CR_{ub}^t(g) = \frac{\left(\sum_{r \in dom\left(R^t(g) \cup P_R^t(g)\right)} \frac{\left(R^t(g)\right)(r)}{\left(R^t(g) \oplus P_R^t(g)\right)(r)} \right)}{\mid R^t(g) \oplus P_R^t(g) \mid} \tag{4}$$

Example 1 *Consider the goal* ExploreRegion(red1) *at the point where* Experiment(rock1) *has completed but neither* Traverse(rock1,canyon) *nor* Survey(canyon) *has started. Let* R^t(Experiment($rock1$)) = {(drill,1),(memory,40),(time,150)}. *Then we have:*

N^t_R(ExploreRegion($red1$)) = {*(drill,0),(memory,120),(time,480)*}, P^t_R(ExploreRegion($red1$)) = {*(drill,1),(memory,150),(time,860)*}, *and further:*
$CR^t_{lb} = (1/(1+0)+40/(40+120)+150/(150+480))/3 = 49.6\%$
$CR^t_{ub} = (1/(1+1)+40/(40+150)+150/(150+860))/3 = 28.6\%$

This approach treats the different resources, e.g., drill bits, memory, fuel, etc., and the resource types, i.e., consumable and reusable, to be of equal importance when measuring completeness, placing the emphasis on domain independence. A future extension would be to weight certain resources or resource types according to domain-dependent criteria. It is straightforward to include these weights in the above computation.

4.3 Effects as a Measure of Completeness

We now turn from resources, a measure of effort, to effects, a measure of accomplishment. As we have discussed, the effects of a goal can be thought of as the state of the world that the agent wants to achieve in order to accomplish the goal. For instance, in the example at the end of Sect. 4.1.2, the rover's goal to survey the area may have the effects of *area-surveyed* and *target-selected*. The percentage of these effects currently achieved gives a quantifiable measure of accomplishment. Note that the issue at hand is not how to express effects (e.g., the language used for $S(g)$) but how to quantify goal completeness. We propose two computational approaches: the first based on the success condition of the goal and the second on the effect summaries of the goal.

4.3.1 Completeness Based on the Success Condition

One way of determining the level of completeness of a goal g, with respect to accomplishment, is to determine the percentage of effects in the success condition $S(g)$ achieved at the current point in time.

In order to compute this measure the agent needs to know the current value of a given effect, and to know the initial values of the effects in the success condition of the goal.

Def. 8 *Let $B^t(e)$ be a function that evaluates the current value α of the effect $e \in \mathcal{E}$ as known by the agent at the current time t.*

Unlike the success condition or effect summaries, where the value of the effect is *what needs to be accomplished* in the future, the value of the effect determined by $B^t(e)$ is the *current* value of the effect e as estimated by the agent.

Def. 9 *The* initial set of effects *for a goal g is $B^i(g) = \{(e, \alpha^i) | e \in \mathcal{E}\}$, where α^i is the value of e when the execution of g begins.*

We compute the level of completeness w.r.t. $S(g)$ by calculating the percentage of the value of each effect in $S(g)$ currently achieved by the agent relative to initial value when the goal execution began and the value to be achieved. For an effect $e \in dom(S(g))$, we write $S(g)(e)$ to denote the value of e in $S(g)$, and similarly for $B^i(g)$.

Def. 10 *The* level of completion *of a goal g at the current time t with respect to the effects in the success condition is:*

$$CE^t_S(g) = \left(\sum_{e \in dom(S(g))} \frac{B^t(e) - B^i(g)(e)}{S(g)(e) - B^i(g)(e)} \right) \Big/ |S(g)| \quad (5)$$

For example, the goal Survey($canyon$) in Fig. 1 has S(Survey($canyon$)) = {TargetList(canyon), Measured(target)}. If IdentifyTargets($canyon$) has completed but Experiment($target$) has not commenced, we have CE^t_S(Survey($canyon$)) = $(1 - 0)/(2 - 0) = 50\%$.

4.3.2 Completeness Based on the Effect Summaries

The above computation does not take into consideration effects other than those in the success condition of the goal, even for those goals where some (side-)effects are necessary in order to achieve the goal's effects. We include these effects as part of the quantification of completeness and use effect summaries to present a lower-bound using the definite effect summary, and an upper-bound using the combined definite and potential effect summaries (since they are exclusive). Note that goal side-effects were also included in the resource summary approach of Sect. 4.2; only in Sect. 4.3.1 are they not relevant.

We adopt the techniques developed in [21] for deriving and updating the effect summaries, but generalize their formulae to operate on a set of effects that are composed of key-value pairs and not simple predicates. This generalization changes the way in which the sets of effects are added ($\oplus$) and merged ($\otimes$). We detail the redefined $\oplus$ operator below but omit the $\otimes$ operator as this work does not use it.

Def. 11 *The* lower-bound effect accomplishment analysis *of a goal* g *at the current time* t, $CE^t_{lb}(g)$, *is:*

$$\left(\sum_{e \in dom(D_E(g))} \frac{B^t(e) - B^i(g)(e)}{D_E(g)(e) - B^i(g)(e)} \right) \Big/ |D_E(g)| \quad (6)$$

Def. 12 *The* upper-bound effect accomplishment analysis *of goal* g *at the current time* t, $CE^t_{ub}(g)$, *is:*

$$\frac{\left(\sum_{e \in dom(D_E(g) \oplus P_E(g))} \frac{B^t(e) - B^i(g)(e)}{(D_E(g) \oplus P_E(g))(e) - B^i(g)(e)} \right)}{|D_E(g) \oplus P_E(g)|} \quad (7)$$

where for any two sets of effect-value pairs E_1 and E_2, $E_1 \oplus E_2 = \{(e, E_1(e) + E_2(e)) \mid e \in (dom(E_1) \cup dom(E_2))\}$

Example 2 *Consider the goal Survey($canyon$) in Fig. 1 which has D_E(Survey($canyon$)) = {TargetList(canyon), ArmPositioned(target), Measured(target)}, and P_E(Survey($canyon$)) = {SpectralProfile(target), ThermalProfile(target)}. If the subgoal IdentifyTargets($canyon$) has completed but Experiment($target$) has not commenced, and none of the effects were true at the start of the goal execution, we have:*
CE^t_{lb}(Survey($canyon$)) $= (1 - 0)/(3 - 0) = 33\%$
CE^t_{ub}(Survey($canyon$)) $= (1 - 0)/(5 - 0) = 20\%$

4.4 Implementation

We have implemented our computational approach, and used it on the example scenario described in Sect. 3. The implementation is in *Orpheus*, a Prolog implementation of the agent language CAN [16]. The additional Prolog code for the computation of the completion measures amounts to approximately 300 lines of code. The implementation processes the goals in the Mars rover scenario in negligible additional time, compared to the time without the extra code for the completeness calculations.

Recall that our emphasis is not on raw computational efficiency, but on finding principled, tractable ways to quantify completion estimates for goals. Hence we do not perform efficiency comparisons with other methods – since there are none to directly compare with, as [20, 14], etc, do not account for partial satisfaction, while [23] do not provide computational mechanisms – nor characterise the likelihood of plan success/failure, but measure goals' progress.

In the scenario, given an imminent deadline such as the approach of dusk, it may be reasonable to terminate the execution of IdentifyTargets($canyon$) once a sufficient fraction of the canyon has been surveyed, or if a sufficient number of targets has been found. This would allow the top-level goal (ExploreRegion($red1$)) to be completed before the deadline, despite not having fully explored the canyon for all possible targets. The information computed at run-time by our approach provides a quantifiable basis for such decisions.

Space limitations prevent us from giving here the detailed traces from the scenario. The code and execution output are available from the authors' website (http://goanna.cs.rmit.edu.au/~jah/orpheus/).

5 Conclusion and Future Work

This work is motivated by how an agent can obtain information to make the most suitable decisions about its courses of action. We have provided a principled mechanism for computing completeness of goals of a BDI-style agent. To our knowledge, this work is the first to study such computation with an emphasis on tractable, pragmatic reasoning. Our technical approach leverages and extends earlier work on efficient resource and effect summarization, and we retain the low computational overhead. An agent can use the estimations of goal completeness in order to inform its deliberation in important areas such as goal prioritization and conflict resolution [23, 20]. This paper provides a foundation for reasoning: it is not our aim here to specify the mechanisms by which an agent exploits this information (see, e.g., the discussion in [23]). The approach in this paper applies to plans as well as goals, except for the completeness based on the success condition (Sect. 4.3.1).

We have implemented the computational approach and used it to analyse a Mars rover scenario. Beyond this scenario, we plan to examine a set of real-world scenarios in order to gain a deeper understanding of the usefulness of the approach, particularly the relative value of the resource-based and the effect-based computations.

Two potential aspects for further work relate to the potentially non-monotonic nature of effects. First, despite having made one or more effects become true, these effects could be undone by either another agent, or by interactions with the environment, such as wind moving rocks around after the agent has positioned its robot arm, or an identified target being moved from its initial location. This means the calculations above would need to take into account the need to re-establish effects which had been previously made true.

Second, one can consider the resource costs for failed plans. For example, if the Mars rover attempts a spectroscopic analysis, but finds that it fails, it may still consume drill bits, memory and time in doing so. This means that we need to adjust the calculations for the definite and potential resource estimates for completing the goal to take the resources used in failed subgoals into account.

Another avenue for further work is to investigate domain-dependent weighting of resource and effect types. For example, if a certain resource is unused (such as the drills resource in the Survey($canyon$) goal in the above example) then its contribution to the goal's overall completeness can be discounted.

A final direction is how to apply the techniques of this paper to more complex goal types, such as *maintenance goals*, for which the agent maintains a given condition, rather than simply achieve it [3]. For such goals which have a more intricate goal life cycle, the challenge is to define a suitable notion of completeness measurement.

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

REFERENCES

[1] B. J. Clement, E. H. Durfee, and A. C. Barrett, 'Abstract reasoning for planning and coordination', *JAIR*, **28**, 453–515, (2007).
[2] M. B. Do, J. Benton, M. van den Briel, and S. Kambhampati, 'Planning with goal utility dependencies', in *Proc. of IJCAI'07*, (2007).
[3] S. Duff, J. Thangarajah, and J. Harland, 'Maintenance goals in intelligent agents', *Computational Intelligence*, **30**(1), 71–114, (2014).
[4] M. Georgeff and A. Rao, 'Rational software agents: From theory to practice', in *Agent Technology: Foundations, Applications, and Markets*, chapter 8, 139–160, Springer, New York, (1998).
[5] B. J. Grosz and L. Hunsberger, 'The dynamics of intention in collaborative activity', *Cognitive Systems Research*, **7**(2–3), 259–272, (2006).
[6] P. Haddawy and S. Hanks, 'Representations for decision theoretic planning: Utility functions for deadline goals', in *Proc. of KR'92*, (1992).
[7] K. V. Hindriks, C. Jonker, and W. Pasman, 'Exploring heuristic action selection in agent programming', in *Proc. of ProMAS'08*, (2008).
[8] R. Holton, 'Partial belief, partial intention', *Mind*, **117**, 27–58, (2008).
[9] Z. Huang and J. Bell, 'Dynamic goal hierarchies', in *Proc. of the 1997 AAAI Spring Symp. on Qualitative Preferences in Deliberation and Practical Reasoning*, pp. 9–17, (1997).
[10] E. Kamar, Y. Gal, and B. J. Grosz, 'Incorporating helpful behavior into collaborative planning', in *Proc. of AAMAS'09*, pp. 875–882, (2009).
[11] S. M. Khan and Y. Lespérance, 'A logical framework for prioritized goal change', in *Proc. of AAMAS'10*, pp. 283–290, (2010).
[12] V. Lesser and et al., 'Evolution of the GPGP/TAEMS Domain-Independent Coordination Framework', *JAAMAS*, **9**(1), 87–143, (2004).
[13] D. N. Morley and K. Myers, 'The SPARK agent framework', in *Proc. of AAMAS'04*, pp. 714–721, (2004).
[14] D. N. Morley, K. L. Myers, and N. Yorke-Smith, 'Continuous refinement of agent resource estimates', in *Proc. of AAMAS'06*, (2006).
[15] R. Rönnquist, 'The goal oriented teams (GORITE) framework', in *Proc. of ProMAS'07*, pp. 27–41, (2007).
[16] S. Sardiña and L. Padgham, 'A BDI agent programming language with failure handling, declarative goals, and planning', *JAAMAS*, **23**(1), 18–70, (2011).
[17] D. E. Smith, 'Choosing objectives in over-subscription planning', in *Proc. of ICAPS'04*, pp. 393–401, (2004).
[18] J. Thangarajah, J. Harland, D. N. Morley, and N. Yorke-Smith, 'Towards quantifying the completeness of BDI goals', in *Proc. of AAMAS'14*, pp. 1369–1370, (2014).
[19] J. Thangarajah, J. Harland, and N. Yorke-Smith, 'A soft COP model for goal deliberation in a BDI agent', in *Proc. of CP'07 Workshop on Constraint Modelling and Reformulation (ModRef'07)*, pp. 61–75, (2007).
[20] J. Thangarajah and L. Padgham, 'Computationally effective reasoning about goal interactions', *J. Automated Reasoning*, **47**(1), 17–56, (2011).
[21] J. Thangarajah, L. Padgham, and M. Winikoff, 'Detecting and avoiding interference between goals in intelligent agents', in *Proc. of IJCAI'03*, pp. 721–726, (2003).
[22] J. Thangarajah, M. Winikoff, L. Padgham, and K. Fischer, 'Avoiding resource conflicts in intelligent agents', in *Proc. of ECAI'02*, pp. 18–22, (2002).
[23] M. B. van Riemsdijk and N. Yorke-Smith, 'Towards reasoning with partial goal satisfaction in intelligent agents', in *Proc. of ProMAS'10*, pp. 41–59, (2010).
[24] M. Winikoff, 'JACK intelligent agents: An industrial strength platform', in *Multi-Agent Programming*, 175–193, Springer, (2005).
[25] Y. Zhou and X. Chen, 'Partial implication semantics for desirable propositions', in *Proc. of KR'04*, pp. 606–612, (2004).
[26] Y. Zhou, L. van der Torre, and Y. Zhang, 'Partial goal satisfaction and goal change: Weak and strong partial implication, logical properties, complexity', in *Proc. of AAMAS'08*, pp. 413–420, (2008).