Iterative Goal Refinement for Robotics

Mark Roberts[1], Swaroop Vattam[1], Ronald Alford[2], Bryan Auslander[3], Justin Karneeb[3], Matthew Molineaux[3], Tom Apker[4], Mark Wilson[5], James McMahon[6], and David W. Aha[5]

[1]NRC Postdoctoral Fellow; Naval Research Laboratory, Code 5514; Washington, DC
[2]ASEE Postdoctoral Fellow; Naval Research Laboratory, Code 5514; Washington, DC
[3]Knexus Research Corporation; Springfield, VA
[4]Exelis Corporation; Alexandria, VA
[5]Navy Center for Applied Research in Artificial Intelligence; Naval Research Laboratory, Code 5514; Washington, DC
[6]Physical Acoustics Branch; Naval Research Laboratory, Code 7130; Washington, DC
[1,2]*first.last*.ctr@nrl.navy.mil | [3]*first.last*@knexusresearch.com | [4]thomas.apker@exelisinc.com | [5,6]*first.last*@nrl.navy.mil

Abstract

Goal Reasoning (GR) concerns actors that assume the responsibility for dynamically selecting the goals they pursue. Our focus is on modelling an actor's decision making when they encounter notable events. We model GR as an iterative refinement process, where constraints introduced for each abstraction layer shape the solutions for successive layers. Our model provides a conceptual framework for robotics researchers and practitioners. We present a goal lifecycle and define a formal model for GR that (1) relates distinct disciplines concerning actors that operate on goals, and (2) provides a way to evaluate actors. We introduce GR using an example on waypoint navigation and outline its application, in three projects, for controlling simulated and real-world vehicles. We emphasize the relation of GR to planning, and encourage PlanRob researchers to collaborate in exploring this exciting frontier.

1. Introduction

Robotic systems often act using incomplete models in environments that are dynamic, partially observable, and non-deterministic. One consequence is that they will encounter notable events. Appropriate responses to notable events can be *designed* a priori or *learned* by the actor. During execution, robots *deliberate* on their responses to notable events and can choose to adjust their expectations or world model, repair their current plan, replan, or regoal (i.e., change their current goal(s)). In each case, they take steps toward achieving goals. We refer to this capability of reasoning about ones goals as *goal reasoning* (GR), which involves dynamically assessing the tradeoffs within the space of goals. We argue that GR is of particular value to robotic systems, as it supports more autonomous behavior.

We present a preliminary formal model that frames GR as an iterative refinement process similar to planning as refinement search (Kambhampati *et al.*, 1995) and iterative repair (Chien *et al.*, 2000). Our model provides a common language for defining GR actors and complements recent foundational perspectives on planning and acting (Ghallab *et al.*, 2014) and deliberation functions (Ingrand & Ghallab, 2014). We present a motivating example (§2), provide background (§3), detail our model and two instantiations (§4), provide a proof of minimal agency (§5), introduce goal memory (§6), define the GR problem (§7), and describe its application to three robotics-related tasks (§8). We integrate our discussion of related work throughout the paper, although this does not constitute a thorough survey on goals in the literature on planning, robotics, agents, and actors.

Our projects span GR actors at the coach, team, and single system levels. Like other research on robotics, a cross-cutting concern is eliciting robotic behavior that is consistent, reliable, trustworthy, verifiable, explainable, and predictable. We invite researchers and practitioners to join our ongoing dialog on the topic of goal reasoning.

2. Waypoint Navigation Example

Figure 1 depicts a simple waypoint navigation task where the robot's goal is `at(y)`. This example could apply to controlling an underwater or micro-aerial vehicle in the context of water currents or wind, respectively, as shown as vectors. Dashed curves indicate the bounds of the expected trajectory (i.e., a soft constraint) while the outer box represents a hard constraint that the actor should avoid violating. The actual trajectory of the robot is given by the solid arc that starts at `x` and ends at `y`. The deviating path is due to the difference between the expected flow (dashed vectors) and actual flow (solid vectors).

From the example it is evident that we assume a dynamic environment, exogenous events, and interruptible actions. We also assume that both the environment and the actor's actions have temporal extent.

Below the plot is a representation of the vehicles timeline, which is inspired by the work of Smith et al. (2000). The time window of the plan indicates that the plan should start executing no earlier than the earliest start time (i.e., the leftmost vertical bar) and finish by the latest finish time (the rightmost vertical bar). The large block in the middle indicates the expected transit duration. Inside the timeline

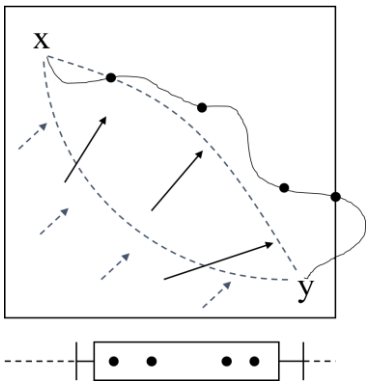


Figure 1: Navigating from `x` to `y`

are dots corresponding to what we call a "notable event," which we define as an event (usually resulting in a state change) that impacts the agent in some way. For this example, the notable events correspond to soft and hard constraint violations; the first two points indicate where the vehicle violates the preferred trajectory while the last two points indicate eminent and actual violation of the hard constraint. These points express possible places and where a decision must be made concerning vehicle behavior.

A GR actor may resolve (i.e., respond to) these events using various strategies (e.g., adjust its expectations, adjust or replace its plan, adjust or change its goal). GR actors differ in the strategies they can apply and how they apply them. In §4, we will use this example to illustrate some of these strategies, after providing some background.

3. Preliminaries

The models and algorithms used for planning in robotics is staggering (e.g., (LaValle, 2006; Ghallab *et al.*, 2014)). We will show how GR can be viewed as a process of refining the constraints on goals. This perspective synthesizes and unifies planning for robotics. First, we define *goals* and review planning as refinement search.

3.1 Goals

Our robotics control applications focus on achieving *goals*, which we define as states an actor desires to achieve (or maintain). To define states, we leverage some elements of the classical planning formalism by Ghallab et al. (2004). Let L be the language for representing world states, $S \subseteq 2^L$ be the set of world states, and $g \subseteq L$ be a goal. Then $S_g = \{s \in S | g \subseteq s\}$ represents the set of states that achieve g. The world is assumed to exist externally to the actor, and a plan is a sequence of actions for transforming an initial state S_0 into S_g. For much of our discussion we focus on a single goal, but the use of a goal set is appropriate for some applications; the model easily extends to goal sets.

We assume that a set of temporal, resource, and ordering constraints apply to goals as well as states and actions, but the exact nature of these constraints is a design decision. Researchers have used a variety of ways to represent such constraints (e.g., as a constraint satisfaction problem (Scala, to appear), in PDDL (Vaquro, to appear), or NDDL (Rajan et al. 2013)). We also assume that completing (or maintaining) goals has intrinsic value for an actor; we return to this assumption in §6 and §7.

For GR to occur the actor must perform actions for transitioning among *both* its external and internal state. To exemplify these, a goal to achieve external state in Figure 1 might be `at(y)`, while a goal to satisfy internal state might be `finished(at(y))`. So we expand and partition the language of the GR actor into $L_{GR} = L_{external} \cup L_{internal}$. We similarly partition the set of goals into $S_g = E_g \cup I_g$. In $L_{external}$ the actor selects actions to achieve E_g. In $L_{internal}$ the actor selects actions to achieve I_g and some internal actions may be conditioned on external goals. *Primitive goals* cannot be decomposed into subgoals.

3.3 Planning as Refinement Search

Our model's theoretical foundation builds from Planning as Refinement Search (Kambhampati, 1994; 1997; Kambhampati *et al.*, 1995), which models planning in a generic way to distinguish planners by their design choices and facilitate their comparison. Refinement planning employs a split and prune model of search, where plans are drawn from a candidate space K. Let a search node N be a constraint set that implicitly represents a candidate set drawn from K. Refinement operators transform a node N_i at layer i into m children $\langle N_{j1}, N_{j2,}, \ldots, N_{jm} \rangle$ at layer $j = i + 1$ by adding constraints which further restrict the candidate sets in the next layer. If the constraints are inconsistent then the candidate set is empty. The authors initially provided two kinds of constraints that can be added by refinement operators: (1) *interval constraints* ensure a variable (i.e., a proposition) maintains a property (i.e., it remains true, false, or unchanged) over an interval; and (2) *truth point constraints* ensure a variable is true at a specific point in time. Kambhampati (1994) conjectured that these constraints could be extended to include behavioral constraints or desires. Kambhampati and Srivastava (1995) extend plan refinement to state-space planning with *contiguity constraints* ensuring, for two actions i and j, no new action can intervene.

Let $N_\emptyset$ represent an initial node whose candidate set equals K and results from only the initial constraint set provided in the problem description (from the perspective of the search process, the refined constraints are empty, thus the subscript $\emptyset$). The REFINEPLAN algorithm begins with $N_\emptyset$ and recursively applies refinements to add constraints until a solution is found. A desirable property of refinements is that each layer of search results in smaller candidates subsets as REFINEPLAN proceeds. Thus the

constraints aid search by identifying inconsistent nodes and providing backtracking points. An optional step is to apply forward consistency checking to further prune candidate nodes, usually at considerable computational cost. Instantiations of REFINEPLAN correspond to different versions of classical planning.

The original model of refinement planning focused on Partial Order Planning, but extensions to the kinds of constraints allowed the refinement framework to incorporate other forms of planning and clarify issues in the Modal Truth Criterion (Kambhampati & Nau, 1994). Unfortunately space limitations prevent a full exposition of these ideas. Briefly, plan refinement allows us to equate different kinds of goal decomposition methods in plan-space and state-space planning. More recent formalisms such as Angelic Hierarchical Plans (Marthi *et al.*, 2008) and Hierarchical Goal Networks (Shivashankar *et al.*, 2013) can also be viewed as leveraging this concept. The focus on constraints in plan refinement also allows a natural extension to the many integrated planning and scheduling systems that use constraints for temporal and resource reasoning.

4. Goal Reasoning as Goal Refinement

To build on plan refinement, we distinguish between a GR process and the actor that is running it because there may be additional processes in the actor such as meta reasoning, learning, etc. We assume a goal is achieved through the execution of some expansion (i.e., plan). A GR process *refines a set of goal nodes G until they can be achieved through execution*. For a goal $g \in G$, a goal node $N^g = \langle C_g, X_g, x, o \rangle$ is a tuple of constraints C_g, possible expansions that could achieve the goal X_g, the currently selected expansion $x \in X_g$, and goal lifecycle mode o. A GR process begins with $N^g_\emptyset$, which consists of the candidate space of all possible executions achieving g, and makes decisions that refine its goals S_g or I_g through a series of refinements R on goal nodes until it selects one expansion $x \in X$ for execution. Similar to plan refinement, goal refinement takes a Least Commitment (Weld, 1994) approach. Further, planning and learning (Veloso *et al.*, 1995) could be incorporated in certain parts of the system, but learning is not a requirement for goal refinement. Defining GR as goal refinement allows us model GR in a generic way to distinguish planners by their design choices and facilitate their comparison

4.1 Constraints (C_g)

We partition the constraints $C_g = C_g^{given} \cup C_g^{added}$ into (1) those given to the GR process from the process that invoked it (e.g., human operator, meta-reasoning process, coach) and (2) those that it adds during refinement.

Constraints can be temporal (finish by a certain time), ordering (do `x` before `y`), maintenance (remain at a certain depth), resource (consider only one goal at a time), or computational (only use so much CPU or memory). Top-level constraints can be pre-encoded or based on drives (e.g., (Coddington *et al.*, 2005; Young & Hawes, 2012)). Hard constraints in C_g must be satisfied at all times (stay within the box in the waypoint example), while soft constraints should be satisfied if possible (follow a preferred trajectory).

4.2 Expansions (X_g)

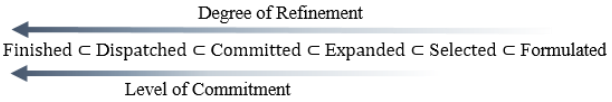
To distinguish goal refinement from planning, we define the set of possible *expansions* X_g as the means of achieving g. A goal S_g or I_g can be achieved by executing an expansion. We call $x \in X_g$ a *selected expansion* that is currently slated for execution.

The term "expansion" highlights GR as any process that performs goal refinement. Planning (in the classical sense) is one kind of expansion, but not all possible expansions of S_g are plans. Robotics systems are often integrated in layers that operate on distinct granular models of the world. So expansions can include, but are not limited to: a simple rule, a richly detailed goal or task network, a (state-based) plan, the switching of behaviors, a change in parameters for an adjoining deliberation layer, an algorithm for learning new or revising existing knowledge, a recipe for unlearning, or, for a team of robots, a specially crafted algorithm.

Expansions can be provided at different times to an actor: *designed* expansions are declared as part of the actor's specification (i.e., as part of L_{GR}), while *learned* expansions are new ways of problem solving derived from the actor's experience and investment in capturing or revising new knowledge (i.e., L_{GR}, which may grow or change to incorporate this new knowledge). This knowledge can be captured online during execution, during the actor's deliberation, or offline prior to execution.

4.3 Modes (o)

The modes of a goal indicate successively smaller candidate sets towards eventual execution. We label each set with a mode from {Formulated, Selected, Expanded, Committed, Dispatched, Finished}. Transitions between these modes lead toward eventual execution (Clement *et al.*, 2007). We present two views of these modes. In the goal refinement view, each mode can be a strict subset of the next in the candidate space of goals:



Reading this from right to left (due to the subset relation): After an actor formulates and selects a goal, planning involves searching through candidate expansions and

selecting one. Finally, the dispatch step dispatches a plan for execution, monitors its trajectory (making corrections as needed), and marks the goal as achieved if execution went well. Each transition to a new mode reduces the candidate set, increases the level of commitment the actor has made to a goal, and increases its degree of refinement. Refinements follow naturally from the view of actors that are performing deliberation (Ingrand & Ghallab, 2014). We invest the next section discussing a left-to-right view of the modes.

4.4 The Goal Lifecycle

The lifecycle for a goal (Figure 2) captures the possible decision points of a GR actor and complements a plan's lifecycle (e.g., (Pollack & Horty, 1999; Myers, 1999). Decisions consist of applying a *strategy* (denoted using an arc in Figure 2; **boldfaced** in this section) that transitions a goal among *modes* (denoted using large or small rounded boxes) in the lifecycle. The *g*'s in the goal lifecycle correspond to goals and *x*'s correspond to expansions. Transitions are verbs and modes qualify a goal's mode (e.g., *select*(S_g) transitions S_g from *formulated* to *selected* mode).

Goals in an *active* mode are those that have been formulated but not yet dropped. The **formulate** strategy is the first decision point. In many actors, this decision is carefully (implicitly, statically) encoded (e.g., an observation may fire a trigger to achieve some goal). Vattam *et al*., (2013) describe goal formulation strategies. The **drop** strategy causes a goal to be "forgotten" and can occur from any active mode. This decision can be sophisticated and involve learning from the execution or attempted expansion of a selected goal. To **select** a goal indicates intent. A selection strategy depends on which goals were formulated and effects the selection of a goal for further refinement. The **expand** strategy decomposes a goal into subgoal(s) or a primitive goal. The **commit**($\boldsymbol{x}$) strategy chooses the best expansion for execution; we assume domain-specific quality metrics can assess expansion quality. The **dispatch**($\boldsymbol{x}$) strategy slates the best expansion for execution and places the goal in an *executing* mode. In both single- and multi-agent systems, a plan may undergo further refinement (i.e., scheduling) prior to execution. Prior refinements could favor a least commitment approach on temporal/resource constraints to allow for flexible dispatch (Conrad *et al.*, 2009).

Because we assume an online dynamic world, goals in an executing mode are subject to transitions that result from expected or unexpected external state changes during execution. The **monitor** strategy can be passive (i.e., do nothing) or proactive (i.e., monitor periodically) while a goal is dispatched. As long as its plan's execution does not encounter any notable events, goal execution follows a normal path toward achievement. If no notable events occur and the dispatched expansion completes, then the **finish** strategy marks the goal as *finished*, which may store the goal to aid future deliberation. Not all goals may reach this mode because goals can be dropped.

When notable events occur, the **evaluate** strategy determines how they impact goal execution (positively or negatively). An *evaluated* mode does not imply that execution of the current expansion is stopped. If the evaluation does not impact the goal, it can **continue** with the execution. However, if the event impacts the current goal, the set of **resolve** strategies define a suite of paths toward goal achievement. An obvious choice is to change the world model using **adjust**($\boldsymbol{L}$), but adjusting its model does not resolve the current goal and further refinements are required. In the waypoint example, we can imagine that the GR actor could decide to adjust the bounds on the preferred trajectory using the variance of the actual path. However, this is only an "internal" adjustment, and it does not subsequently communicate the adjusted expectations to its execution system. The GR would simply ignore any future bounds violations that fall within the new expectations. Rather than adjust its expectations, the GR might apply a **repair**($\boldsymbol{x}$) strategy by repairing $\boldsymbol{x}$ so that it meets the new context; this is the so-called replanning approach. In the waypoint example, this might involve communicating new bounds, allowing more time, or selecting behaviors more appropriate to the current conditions. This resolution repairs but does fundamentally alter the current plan. If no repair is possible (or desired) then the GR can apply the **re-expand**($\boldsymbol{g}$) strategy to reconsider the original plan from scratch. In the

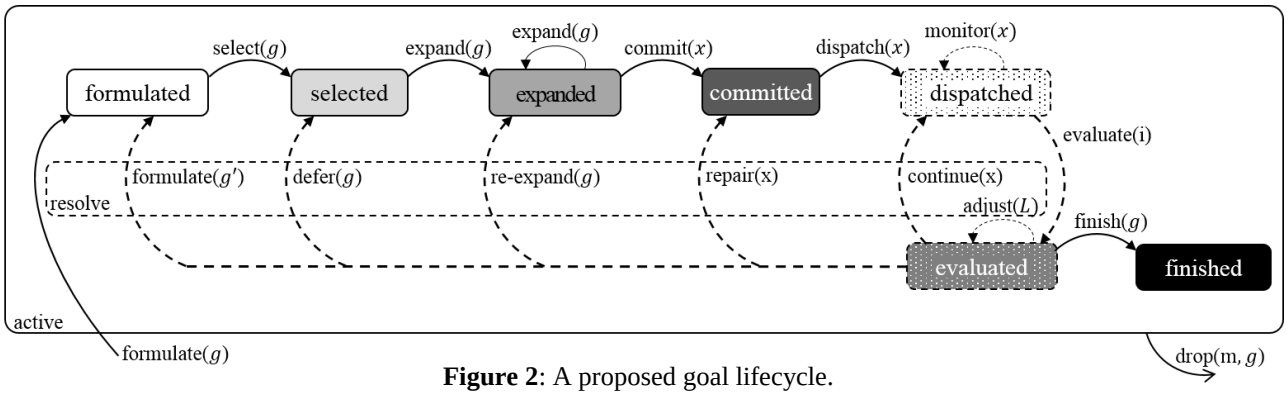


Figure 2: A proposed goal lifecycle.

waypoint example, this might occur if an obstacle is found to exist en route. The **defer**(g) strategy instead postpones the goal for further processing. In the example, this may happen if current conditions are deemed as unfavorable for achieving g, but the GR decides to retain g as worth pursuing in the future. A final option occurs in **formulate**(g'), which abandons g in favor of a newly formed goal g'.

We very recently discovered the work on goal lifecycles in the Autonomous Agents literature that is closely related to the goal lifecycle proposed in this paper. Harland et al. (2014) extend their earlier work (Thangarajah et al., 2010) to present a goal lifecycle for BDI agents, provide operational semantics for their lifecycle, and demonstrate the lifecycle on a Mars rover scenario. Work by Winicoff et al. (2010) has also linked Linear Temporal Logic to the expression of goals that the project we discuss in §8.3 may be able to leverage. We plan to fully explore these approaches in future work.

4.5 Strategies and Strategy Composition

Strategies can be simple or complex. For example, *select*(g) could be implemented as a simple rule (e.g., automatically select any goal that is formulated) or it could be implemented as a learned policy that considers knowledge about the environment, which goals are currently executing, and their priority. Similarly, the *drop* strategy could be very simple (e.g., in Figure 1, drop any goal when hard constraints are violated), or the *drop* strategy could attempt to learn long-term knowledge from the information gathered in the goal node as it transitioned through the life cycle (e.g., in Figure 1, adjust future expectations for this region to account for greater flow).

Strategies can be composed, of which the *resolve* strategy in Figure 2 is one example. A composition representing classical planning demonstrates goals that are formulated by an external process to the actor. Let g_2 be a goal and FINDPLAN be a planning algorithm producing one expansion x that equates to a plan for achieving g_2. Then a classical planning strategy is composed:

$$\textit{plan}(g_2) \Rightarrow \text{formulate+select}(g_2),\ x = \text{FINDPLAN}(g_2).$$

Another strategy composition can relate the goal lifecycle to partial satisfaction planning (PSP) and soft goals for planning (Benton et al., 2010). PSP is closely aligned with GR for the project discussed in §8.2. In that framework planning proceeds on an *oversubscribed set* of goals where a penalty is assessed for not meeting goals; the planner constructs a plan that maximizes the utility (i.e., the payoff) while minimizing the penalty. Let G_3 be an oversubscribed goal set and PSPPLAN be a planning algorithm that selects the subset of goals and a plan x to achieve them given the objective criteria. Then a PSP planning strategy is composed:

$$\textit{psp-plan}(G_3) \Rightarrow \text{formulate}(G_3),\ x = \text{PSPPLAN}(G_3)$$

It may seem *psp-plan*(G_3) is identical to *plan*(g_2), however PSPPLAN(G_3) performs goal selection, expansion and plan selection, while FINDPLAN (g_2) only performs expansion and plan selection.

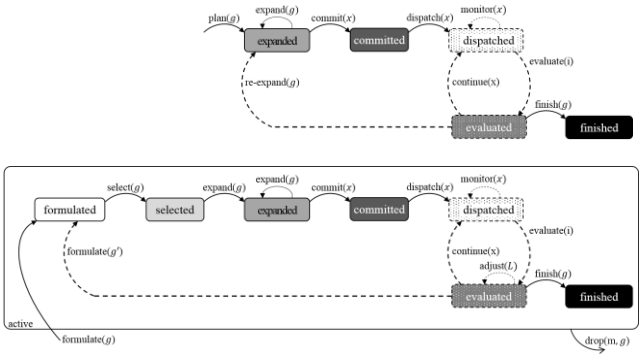


Figure 3: Contrasting replanning (top) with GDA (bottom) instantiations of the GR model.

4.6 Instantiations of Goal Reasoning

We next demonstrate how the full GR model can instantiate two existing GR systems. The first is a replanning system, which is a common approach to solving online dynamic planning (e.g., (Yoon *et al.*, 2007)). Figure 3 (top) shows how the GR model can instantiate such replanning systems. As shown, formulation is presumed (designed) in this model and we have used the "plan(g)" composition of §4.5.

Figure 3 (bottom) shows one instantiation of the GR model for Goal-Driven Autonomy (GDA); GDA agents perform online planning and execution (Klenk *et al.*, 2013). A GDA agent consists of an intelligent controller that not only interacts with a planner and the execution environment, but also includes components for GR, separating the planning process from those for goal formulation and management. The controller takes as input an initial planning problem and sends it to the planner. The planner returns (1) a plan, which is a sequence of actions, and (2) a corresponding sequence of expectations consisting of the states expected to result after executing each action in the plan. The controller *dispatches* the plan's actions to the execution environment and then runs a sophisticated *evaluate* strategy. While a plan is *dispatched*, the controller *evaluates* new information by (1) comparing observations with expectations to discover discrepancies. For a discrepancy, it (2) may generate an explanation, which takes the discrepancy plus a history of past actions and observations to propose one or more possible causal

explanations. Depending on the explanation, the controller (3) may *adjust(L)* to correct its model or expectations of the world, *formulate* a new goal in response to the discrepancy, and then *drop* the existing goal. Finally, (4) this new goal moves through the goal lifecycle (*selected*, *expanded*, etc.) and is weighed against other goals for execution.

4.7 Instantiations of Strategies

We now detail two possible instantiations of the strategies for the GDA architecture from Figure 3 (bottom). The M-ARTUE system (Wilson et al, 2013) takes a direct approach to the goal formulation step, in which all possible goals are considered by the agent according to a set of domain-independent heuristics that assess a goal's fitness in three different dimensions: social, exploration and opportunity. The fitness of each goal in these three dimensions is weighted by the urgency of each of the respective needs to come up with a single score by which all goals are compared. A preliminary study of M-ARTUE showed that it reached performance comparable to the use of domain-specific knowledge for guiding goal selection in a test domain.

The FoolMeTwice agent (Molineaux & Aha, to appear) employs a monitor and integrate strategies that incorporate the environment to effect the agent's knowledge of what events are possible. The FoolMeTwice agent is "surprised" when it cannot explain (i.e., find a history of events consistent with) its observations of the environment. When surprised, FoolMeTwice hypothesizes a new model of a previously unknown event that caused its surprise. These models are thereafter use both monitor and integrate strategies to improve the agent's performance at these tasks.

5. A Proof of Minimal Agency

Decision is a key element of agency. The degree to which the actor's decisions depend on dynamic deliberation rather than pre-encoded knowledge determines its *agency*. Clearly, if a single rule (i.e., a design) covers all contingencies for any strategy that arises during execution, then the actor need not make a decision. Decision points are noteworthy because – assuming the agent acts on a single external goal E_g – the actor's goal switches from achieving E_g to achieving some internal decision goal $I_d \in I_g$. Agency is an increasing function of the number and kind of decision goals. Further, we can prove:

Proposition 1: *The number of primitive active goals for a deliberative agent must be at least two, one of which must be a decision goal, I_d.*

Proof sketch: We can reason about any transition in the lifecycle without loss of generality. Suppose the actor must decide on one of two possible paths for $select(S_g)$ for a primitive goal S_g. There are two cases to consider regarding whether the agent deliberates on this decision. Let a *rule* be a provided (or learned) sequence of steps to achieve some goal. (1) Suppose the actor applies a rule to decide (even non-deterministically), then the agent did not deliberate, which contradicts our assumption of a deliberative actor. (2) If there is no such rule, then the agent must formulate an internal decision goal $I_d =$`is-selected-`S_g (i.e., the mode of S_g is *selected*) and switch to the goal `is-achieved-`I_d. We can imagine many different ways the actor might achieve this new goal (e.g., it can explore to generate knowledge or exploit its existing knowledge). At this point, we have already shown that at least two primitive goals were required. *Q.E.D.*

Corollary 1: *An actor will be unable to resolve a decision goal if it tries to deliberate without enough designed (or learned) knowledge.*

Corollary 2: *The number, kind, and frequency of primitive decision goals defines a spectrum of deliberation and agency for actors.*

6. Goal Memory

In addition to moving goals through a lifecycle toward achievement, a decision-making actor assesses tradeoffs between various criteria (e.g., priority, domain-specific quality functions, global utility, long term payoff). We use the goal lifecycle in conjunction with a *goal memory* to characterize a goal management process that simultaneously addresses both.

Our use of the term *goal memory* here is distinct from its typical use in cognitive goal memory. In cognitive science, goal memory is typically discussed as a mental construct with representations and processes that are used to store and manage goal-related requirements of the task that a cognitive agent happened to be engaged in. While issues such as interference level, strengthening and priming constraints are key requirements to mimic human memory (Altmann & Trafton 2002), we ignore any such considerations because we are not concerned with the cognitive plausibility of our model of the goal memory.

Figure 4 shows the $m \times n$ goal memory in a table M, where a cell M_{ij} represents the i^{th} goal g_i $(1 \leq i \leq m)$, its j^{th} quality criteria $q_i (1 \leq j \leq n)$, and mode. We describe these criteria in the following paragraphs.

The *priority* criterion of a goal determines its importance relative to other goals. Let $f_{priority}(g_i)$: $g_i \rightarrow \mathbb{I}$, then $M_{i1} = f_{priority}(g_i)$. Priority depends on the agent's current state, and so may change dynamically.

	Priority	Inertia	Mode	Quality Metrics		
	q_1	q_2	q_3	q_4	...	q_n
g_1						
g_2						
...						
g_m						

Figure 4*:* A Goal memory of m goals and n quality metrics.

The *inertia* criterion of a goal g_i characterizes the strength of bias against changing its current mode because of prior commitments. Let $f_{intertia}(g_i) : g_i \rightarrow \mathbb{I}$, and let $M_{i2} = f_{inertia}(g_i)$. Inertia is defined as a function of g_i's mode, its number of expansions $|X_g|$, its *staleness* (i.e., the number of time steps since it last transitioned), the number of transitions that have been applied to g_i, and the resources committed to g_i.

The *mode* criterion of a goal g_i determines its relative importance based on how far along it is in the goal lifecycle. For instance, if a goal is closer to execution, it has a higher value because we want to move goals toward finished. Let $f_{mode}(g_i)$: $g_i \rightarrow \mathbb{I}$, then $M_{i3} = f_{mode}(g_i)$.

The remaining criteria express the quality (e.g., cost, value, risk, reward) of achieving g_i with the currently selected expansion $x \in X_g$. These are domain-specific quality metrics, and we provide some examples when discussing applications in §8. These metrics may also include domain-independent quality metrics such as minimizing makespan (i.e., parallel execution time) or minimizing the plan length (i.e., number of plan steps).

In addition to the aforementioned criteria, we conjecture that additional book-keeping columns may be necessary. These include but are not limited to constraints, alternative expansions, parent of goal, type of goal, etc.

7. The Goal Reasoning Problem

A GR agent examines its goal memory state M_t at time t and chooses a strategy that maximizes its long-term rewards using $\sum_t^\infty \gamma^t rew_t$, where γ^t is a discount factor and rew_t is the agent's reward at t, which we model as $rew: M \times R \rightarrow \mathbb{R}$.

For our formal model, we make some simplifying assumptions that are too limiting for integrated robotics but aid in explaining the model. In addition to the dynamic environment and interruptible actions already assumed, we assume for the exposition of the model:

- **Markovian dynamics:** The current choice for an actor is based only on its last (known) state.
- **Infinite horizon with discount:** The actor is myopic; it considers distant future states to be less important than eminent future states, and may favor locally optimal solutions that are globally suboptimal.
- **Non-deterministic actions**: An actor's action may have multiple possible outcomes.

Under these assumptions, we can model GR as a *Markov Decision Process* (MDP), given that the transition function and the reward function are known. Given a current goal memory state M, a GR strategy $r \in R$ and a next goal memory state M', the mode transition function $T(M, r, M')$ is the probability of transitioning from M to M' using strategy r. For MDPs, there exists an optimal deterministic stationary policy (Kaelbling *et al.*, 1996), implying the existence of an optimal value function for a current goal memory state M:

$$V^*(M) = max_{policy}\left(E\left(\sum_{t=0}^{\infty} y^t\, rew_t\right)\right),$$

where $(0 \leq \gamma < 1)$ is the discount factor. This optimal value function is unique and reduces to $\forall M' \in r(M)$:

$$V^*(M) = max_r(rew(M,r) + \gamma \sum_{M'} T(M,r,M')V^*(M')).$$

Given this, we can specify the optimal policy as:

$$policy^*(M) = \arg max_r\,(rew(M,r) + \gamma \sum_{M'} T(M,r,M')V^*(M')).$$

If T or rew are unknown, then GR can be modeled as a reinforcement learning (Sutton & Barto, 1998) problem, where deliberation results in a learned policy. Reinforcement learning is a rich area of research that is out of scope for this paper.

8. Applications of Goal Reasoning

Our group is working on two robotics and one simulated robotics projects involving GR. We review these with a focus on the expected value added by using GR, our technical approach, and the GR research questions we are addressing.

8.1 Unmanned Underwater Vehicle (UUV) Control

UUVs have been used for tasks such as inspection of underwater structures (Antonelli et al., 2001), mine countermeasures (LePage & Schmidt, 2002), and scientific observation (Binney et al., 2010). These have engendered work on motion planning (e.g., Tan et al., 2004), which can guide vehicles to desired locations but cannot select goals. These missions have short duration (at most eight to sixteen hours) and operate over a small region.

Long-duration missions, potentially lasting weeks or months over much larger regions, present new challenges for guidance systems, as the ocean environment is unpredictable and partially observable. A UUV on a long-duration mission must react competently to notable objects and events. It may need to change its objectives or even abort its mission due to unforeseen environmental hazards, underwater barriers, encounters with other vehicles, or

failures of onboard systems. A common approach in the face of a dynamic environment would be replanning. Cashmore et al (2013) confront the need for long-duration autonomy in UUVs and examine the problem of modeling motion for task-level mission planning. Their architecture reacts to notable events (observations of the environment that differ from assumptions) by remodeling the environment and replanning for a fixed set of goals. In the language of our GR model, their approach applies the *adjust(L)* strategy followed by the *re-expand(x)* strategy. This is a case where *re-expand(x)* equates to replanning.

An alternative approach could allow the UUV to *regoal*. Consider a UUV taking oceanographic measurements (e.g., water salinity) when it detects a nearby surface vessel. While motion planning systems will likely continue the measurement task, minimizing risk of collision while maximizing data quality, they cannot consider the broader implications of the vessel's arrival and how best to respond. Depending on the location, nature of the mission, and the identity of the approaching vessel, the UUV may need to communicate with it, attempt to avoid detection, or abort the data-collection mission and return to notify its operator of the surface vessel's approach. An at-sea UUV has limited communication with human operators, and must make such goal decisions autonomously.

To provide a UUV with the ability to reason about and dynamically select goals while pursuing long-term missions, we are applying GDA to guide a Bluefin underwater vehicle, initially in simulation but with planned execution on a real vehicle. GDA can generate appropriate goals in response to unplanned situations and is therefore well-suited to the control of unmanned vehicles at sea.

We use MOOS-IvP (Benjamin et al., 2010) to provide reactive navigation guidance. MOOS is a message-passing system with a centralized publish-subscribe model. IvP Helm is a behavior-based MOOS application that chooses desired heading, speed, and depth for the vehicle in a reactive manner to generate collision-free trajectories. IvP Helm uses an interval programming technique that optimizes over an arbitrary number of objective functions to generate desired navigation values. The GDA agent complements the IvP Helm's reactive behaviors by enabling the capacity for deliberative reasoning for longer missions. Thus, we use the GDA agent to perform GR, IvP Helm to provide navigation guidance, and Bluefin's Huxley control architecture for low-level control.

8.2 Unmanned Air Vehicle (UAV) Control

UAVs have been used frequently in military operations, controlled via teleoperation in surveillance and targeting missions, for example. As they become more autonomous they will also be deployed in air combat operations in areas that are highly dynamic, uncertain, and adversarial. In such environments, UAVs will have to coordinate with manned aircraft, which the USA military highlights as a critical technical challenge (DoD, 2013). Our project's objective is to develop and demonstrate the utility of a GR agent for controlling simulated UAVs in manned-unmanned air combat teams, where the teamed pilots will manage the UAVs' activities.

The air combat environment is highly complex with stochastic, dynamic, adversarial, and partially observable elements. Highly autonomous decision making in such an environment requires agents to respond to situations for which they lack pre-programmed responses. The UAVs cannot rely solely on the pilot for constant oversight in these situations because they must pilot their own vehicle.

For this task, we are integrating a novel GR agent in a decision-making system called the TBM (Tactical Battle Manager), which should advance the state-of-the-art in several respects. This high-tempo environment requires decisions to be made within seconds as indecision could lead to loss of human life or destruction of expensive assets. The human pilot must specify goals and preferences. Finally, scenarios will consist of multiple vehicles, and actions by the actor effect the pilots and other UAVs' agents.

Our GR model is inspired by Young and Hawes's (2012) model, in which *desires* are satisfied via a system of *drives*. At any given time the desire monitor and state monitor inspect the world state. If an event agitates a desire, then the GR may formulate a new goal. For example, suppose a manned vehicle in a manned-unmanned team of vehicles was just shot down. This event would agitate an ENSURE HUMAN SAFETY desire in the actor controlling the UAV, and a drive would then formulate a DEFEND CRASH SITE goal.

To evaluate GR we will use two modern air combat simulations, namely the Next Generation Threat System (NGTS) (2013) and the Analytic Network for Network-Enabled Systems (AFNES). We have integrated a simple GR agent with NGTS; it replaces plans to control air vehicles when notable events occur. We will apply GR to a set of simulated scenarios (with random variations) and measure mission success with and without goal reasoning.

We hypothesize that the integration of GR in TBM will increase the performance of mixed teams in air combat missions, and that GR will reduce the amount of oversight that pilots must provide to their UAV teammates, because they will be able to reason and respond to unexpected situations as they occur.

8.3 Control for Collaborative Sensing

Between the time of a tragic disaster (e.g., the Philippines Typhoon) and the arrival of support operations, emergency response personnel need information concerning the whereabouts of survivors, the condition of infrastructure, a suggested ingress and evacuation routes. Current practice

for gathering this information relies on drone operators and human pilots of helicopters. We believe a hetealrogeneous team of autonomous vehicles with sensor platforms can automate many parts of the information gathering, thus freeing humans to perform more critical tasks and improving the response time for Humanitarian Assistance/Disaster Relief operations.

Planning trajectories for teams *a priori* to achieve a single objective requires solving a high dimensional optimization problem (Yilmaz *et al*., 2008) to compute optimal trajectories that are tightly coupled to the initial assumptions/goal. Bio-inspired and other reactive guidance strategies simplify this problem by using more goal-directed behaviors for area coverage (Liu & Hedrick, 2011) and discrete target tracking (Haque *et al.*, 2008; Kruecher *et al.*, 2007). These behaviors rely on local measurements and instantaneous gradients to guide robots. Still, no behavior or trajectory can handle all contingencies.

A promising approach, inspired by animal behavior, uses finite state automata (FSA) for mobile robot guidance (Balch *et al.*, 2006). Hand-coding an FSA for each execution of a robot is tedious and error prone. Kress-Gazit et al. (2009) instead synthesize an FSA using a Linear Temporal Logic specification (LTL-spec) that views synthesis as searching for a game-theoretic table in which the robot takes actions to achieve its goals against actions taken by the environment (i.e., the adversary). This strategy guarantees correct behavior if the LTL-spec is never violated, but synthesis is exponential in the number of (environmental and sensing) goals. This is clearly intractable for large teams of robots, and we use GR as a "coach" to select goals to maintain tractable synthesis for individuals within the team.

Our technical approach draws on the goal refinement process of decomposing a high-level goal into predicates that then guide LTL-to-FSA synthesis. As the potential number of predicates is large for a multi-vehicle, multi-goal mission, we developed a hierarchical approach that separates GR, planning/scheduling, and vehicle guidance tasks into discrete processes. Our approach converts active goals into an LTL specification for the mobile agents. If a specification is unsatisfiable, an error report is returned to the GR actor.

At runtime, we monitor the agents' progress through their FSAs, and constantly update the GR actor's model of the environment. Notable events occur when the actors: achieve a substantial sub-goal; determine they cannot achieve their current sub-goal; or their FSA does not specify how to respond to an unexpected change in a vehicle's state. Candidate goals are evaluated using an approximate environment and team model based on the MASON (Luke *et al.*, 2005) multi-agent simulator, which can perform some FSA synthesis and apply the optimal or reactive guidance algorithms. These produce the quality metrics the GR agent uses to select which goals to activate.

9. Summary and Future Work

We observed that goal reasoning (GR) occurs when an actor observes notable events and that it falls along a spectrum of design to deliberation. The extent to which an actor takes initiative to deliberate over its goals provides a measure of autonomy. We presented the goal lifecycle to demonstrate how modes act as constraints in the management of goals and discussed how this lifecycle instantiates GR models: for replanning and Goal-Driven Autonomy (GDA). We formalized the GR problem by introducing a goal memory and GR operators, casting the problem of GR in terms of choosing a composition of GR operators to maximize an actor's future rewards. Although our model is general enough to be agnostic about which approach is used for this purpose, we then related the GR problem to a Markov Decision Process (if states and the value function are known) and Reinforcement Learning (if states or the value function are unknown). Finally, we discussed three ongoing robotics-related projects in which we are using a model of GR for decision making and control.

There are many benefits to our proposed GR model:

a. It provides a *common language* for discussing deliberation in actors, and is rich enough to frame the conversation among researchers who study robotics, planning, or scheduling.
b. It is *instantiable*; it covers existing subclasses and can grow to future knowledge/systems.
c. Strategies make the model *composable* and able to incorporate the variety of design decisions of an actor. Strategies can be empty (i.e., no-op), static/dynamic policies, hand-coded (or learned) rules or cases, or domain-specific algorithms.
d. From a software design perspective, the model allows for *rapid prototyping* of systems. A team can begin with handcoded/no-op strategies to determine a platform's viability, which provides a baseline for assessing autonomy. This low-bar approach also aids in focusing knowledge modeling on only the parts of the system where decisions will be made, and thus helps with knowledge engineering for robotics.
e. This model *spans layers of deliberation* at the individual, team, and coach levels.

We will soon extend the formal model of GR, instantiate it in the projects described in §8, and analyze its advantages and limitations by evaluating those actors' performances. We are especially interested in linking the GR lifecycle to recent models of replanning (Talamadupala et al., 2013) and continual planning (Scala, to appear).

We described goal reasoning in terms of a lifecycle that refines an actor's goals and expansions, and summarized its application to robotics-related tasks. Our research is in its early stages, and we invite feedback on this model.

Acknowledgements

The authors for this project were funded by OSD. We also thank the anonymous reviewers whose comments helped improve the paper.

References

Altmann, E. M., & Trafton, J. G. (2002). Memory for goals: An activation-based model. *Cognitive Science*, 26, 39-83.

Antonelli, G., Chiaverini, S., Finotello, R., & Schiavon, R. (2001). Real-time path planning and obstacle avoidance for RAIS: Aan autonomous underwater vehicle. *IEEE Journal of Oceanic Engineering*, *26*(2), 216-227.

Balch, T., Dellaert, F., Feldman, A., Guillory, A., Isbell, C.L., Khan, Z., Pratt, S.C., Stein, A.N., & Wilde, H. (2006). How multirobot systems research will accelerate our understanding of social animal behavior. *Proceedings of the IEEE*, *94*(7), 1445-1463.

Benton, J., Do, M., & Kambhampati, S. (2009). Anytime heuristic search for partial satisfaction planning. *Artificial Intelligence*, *173*(5-6), 562–592.

Benjamin, M., Schmidt, H., Newman, P., & Leonard, J. (2010). Nested autonomy for unmanned marine vehicles with MOOS-IvP. *Journal of Field Robotics*, *27*(6), 834-875.

Binney, J., Krause, A., & Sukhatme, G.S. (2010). Informative path planning for an autonomous underwater vehicle. In *Proceedings of the 2010 IEEE International Conference on Robotics and Automation*, (pp. 4791-4796). Anchorage, AK: IEEE Press.

Chien S., Knight R., Stechert A., Sherwood R., and Rabideau, G. (2000) Using Iterative Repair to Improve the Responsiveness of Planning and Scheduling. *Proceedings of the Conference on Automated Planning and Scheduling* (pp. 300-307). Menlo Park, CA: AAAI Press.

Cashmore, M., Fox, M., Larkworthy, T., Long, D., and Magazzeni, D. (2013). Planning Inspection Tasks for AUVs. In *Proceedings of MTS/IEEE OCEANS 2013*. San Diego, CA: IEEE Press.

Clement, B.J., Durfee, E.H., & Barrett, A.C. (2007). Abstract reasoning for planning and coordination. *Journal of Artificial Intelligence Research*, *28*, 453–515.

Coddington, A.M., Fox, M., Gough, J., Long., D., & Serina, I. (2005). MADbot: A motivated and goal directed robot. *Proceedings of the Twentieth National Conference on Artificial Intelligence* (pp. 1680-1681). Pittsburgh, PA: AAAI Press.

Conrad, P., Shah, J. & Williams, B. (2009) Flexible execution of plans with choice. *Proceedings of the Conference on Automated Planning and Scheduling* (pp. 74-81). Menlo Park, CA: AAAI Press.

DoD (2013). Unmanned systems integration roadmap: FY2013-2038 (Reference Number 14-S-0553). Department of Defense, Washington, DC.

Ghallab, M., Nau, D.S., & Traverso, P. (2004). *Automated planning: Theory and practice*. San Mateo, CA: Morgan Kaufmann.

Ghallab, M., Nau, D., & Traverso, P. (2014). The actor's view of automated planning and acting: A position paper. *Artificial Intelligence*, *208*, 1–17.

Harland, J., Morley, D., Thangarajah, J., & Yorke-Smith, N. (2014). An operational semantics for the goal life-cycle in BDI agents. *Autonomous Agents and Multi-Agent Systems*, *28*(4), 682–719.

Haque, M., Rahmani, A, & Egerstedt, M. (2010). Geometric foraging strategies in multi-agent systems based on biological models. In *Proceedings of the 49th IEEE Conference on Decision and Control*. Atlanta, GA: IEEE Press

Ingrand, F., & Ghallab, M. (2014). Robotics and artificial intelligence: A perspective on deliberation functions. *AI Communications*, *27*(1), 63-80.

Kaelbling, L.P., Littman, M.L., & Moore, A.P. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4, 237-285.

Kambhampati, S. (1994). Design tradeoffs in partial order (plan space) planning. *Proceedings of the Second International Conference on Artificial Intelligence Planning Systems* (pp. 67-97). Chicago, IL: AAAI Press.

Kambhampati, S. (1997). Refinement Planning as a unifying framework for plan synthesis. *AI Magazine*, *18*(2), 67-97.

Kambhampati, S., Knoblock, C.A., & Yang, Q. (1995). Planning as refinement search: A unified framework for evaluating design tradeoffs in partial-order planning. *Artificial Intelligence*, *76*, 168-238.

Kambhampati, S. & Nau, D. (1994). On the nature of modal truth criteria in planning. *Proceedings of the 12th National Conference on Artificial Intelligence* (pp. 67-97). Seattle, WA: AAAI Press.

Kambhampati, S., & Srivastava, B. (1995). Universal classical planner: An algorithm for unifying state space and plan space planning. *New Directions in AI Planning* (pp. 261-271). IOS Press.

Klenk, M., Molineaux, M., & Aha, D.W. (2013). Goal-driven autonomy for responding to unexpected events in strategy simulations. *Computational Intelligence*, *29*(2), 187-206.

Kress-Gazit, H., Fainekos, G.E., & Pappas, G.J. (2009). Temporal logic based reactive mission and motion planning. *IEEE Transactions on Robotics*, *25*(6), 1370-1831.

Kreucher, C.M., Hero, A.O., Kastella, K.D., & Morelande, M.R. 2007. An Information-Based Approach to Sensor Management in Large Dynamic Networks. *Proceedings of the IEEE* 95(5), 978-999

LaValle, S., M. (2006). *Planning Algorithms*, Cambridge University Press.

LePage, K.D., & Schmidt, H. (2002). Bistatic synthetic aperture imaging of proud and buried targets from an AUV. *Journal of Ocean Engineering*, *27*(3), 471-483.

Liu, S-Y., & Hedrick, J.K. (2011). The application of domain of danger in autonomous agent team and its effect on exploration efficiency. In *Proceedings of the IEEE American Control Conference.* San Francisco, CA: IEEE Press.

Luke, S., Cioffi-Revilla, C., Panait, L., Sullivan, K., & Balan, G. (2005). Mason: A multiagent simulation environment. *Simulation*, *81.7*(2005), 517-527.

Marthi, B, Russell, S., & Wolfe, J. (2008). Angelic hierarchical planning: Optimal and online algorithms. *Proceedings of the International Conference on Automated Planning and Scheduling* (pp. 222-231). Menlo Park, CA: AAAI Press.

Molineaux, M., and Aha, D.W. (to appear). Learning Unknown Event Models. In *Proceedings of the Twenty-Eighth AAAI Conference*. Quebec City, Quebec, Canada.

Myers, K.L. (1999). CPEF: A continuous planning and execution framework. *AI Magazine*, *20*(4), 63-69.

NGTS (2013). Next Generation Threat System. [www.navair.navy.mil/nawctsd/Programs/Files/NGTS-2013.pdf]

Pollack, M.E., & Horty, J. (1999). There's more to life than making plans: Plan management in dynamic, multiagent environments. *AI Magazine, 20*, 71-83.

Rajan, K., Py, F., & Barreiro, J. (2013). Towards deliberative control in marine robotics. In *Marine Robot Autonomy* (pp. 91–175). Springer.

Scala, E. (to appear). Continual planning via reconfiguration and goal revision. In *Working notes of the ICAPS Workshop on Planning and Robotics*.

Shivashankar, V., Alford, R., Kuter, U., & Nau, D. (2013). The GoDeL planning system: A more perfect union of domain-independent and hierarchical planning. *Proceedings of the 23rd International Joint Conference on Artificial Intelligence* (pp. 2380-2386). Beijing, China: AAAI Press.

Smith, D., Frank, J., & Jonsson, A. (2000). Bridging the gap between planning and scheduling. *Knowledge Engineering Review, 15*, 61-94.

Sutton, R.S., & Barto, A.G. (1998). *Introduction to reinforcement learning*. Cambridge, MA: MIT Press.

Talamadupula, K., Smith, D. E., Cushing, W., & Kambhampati, S. (2013). A Theory of Intra-Agent Replanning. *Working notes of the ICAPS Workshop on Distributed Multiagent Planning*.

Tan, C.S., Sutton, R., & Chudley, J. (2004). An incremental stochastic motion planning technique for autonomous underwater vehicles. In *Proceedings of IFAC Control Applications in Marine Systems Conference* (pp. 483-488). Ancona, Italy: Elsevier.

Thangarajah, J., Harland, J., Morley, D., & Yorke-Smith, N. (2011). Operational behaviour for executing, suspending, and aborting goals in BDI agent systems. In *Declarative Agent Languages and Technologies VIII* (pp. 1–21). Toronto, Canada: Springer.

Tan, C.S., Sutton, R., & Chudley, J. (2004). An incremental stochastic motion planning technique for autonomous underwater vehicles. In *Proceedings of IFAC Control Applications in Marine Systems Conference* (pp. 483-488). Ancona, Italy: Elsevier.

Vattam, S., Klenk, M., Molineaux, M., & Aha, D. W. (2013, December). Breadth of Approaches to Goal Reasoning: A Research Survey. In *Goal Reasoning: Papers from the ACS Workshop* (p. 111).

Vaquero, T., Nejat, G., & Beck, J.C. (to appear). Planning and scheduling single and multi-person activities in retirement home settings for a group of robots. In *Working notes of the ICAPS Workshop on Planning and Robotics*.

Veloso, M., Carbonell, J., Perez, A., Borrajo, D., Fink, E., & Blythe, J. (1995). Integrating planning and learning: The PRODIGY architecture. *Journal of Experimental & Theoretical Artificial Intelligence*, *7*(1), 81-120.

Weld, D.S. (1994). An introduction to least commitment planning. *AI Magazine, 15*, 27-61.

Wilson, M., Molineaux, M., & Aha, D.W. (2013). Domain-Independent Heuristics for Goal Formulation. In *Proceedings of the Twenty-Sixth International Florida Artificial Intelligence Research Society Conference*. St. Pete Beach, Florida.

Yilmaz, N.K, Evangelinos, C., Lermusiaux, P., & Patrikalakis, N.M. (2008). Path planning of autonomous underwater vehicles for adaptive sampling using mixed integer linear programming. *IEEE Journal of Oceanic Engineering*, *33*(4), 522–537.

Yoon, S.W., Fern, A., & Givan, R. (2007). FF-Replan: A baseline for probabilistic planning. *Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling* (pp. 352-359). Providence, RI: AAAI Press.

Young, J., & Hawes, N. (2012). Evolutionary learning of goal priorities in a real-time strategy game. In *Proceedings of the Eighth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*. Stanford, CA: AAAI Press.