Plan Execution, *Part (I)*: The Individual Agent

Gal A. Kaminka

Computer Science Department and Gonda Brain Research Center
Bar Ilan University, Israel

Abstract

A common approach to building robot control systems is algorithmic in nature; it presupposes that robots are tools to be manipulated perfectly by an algorithm built specifically to optimally solve a given task. But the complexities of realistic tasks and missions in real-world environments are not easily addressed by this approach. This paper posits that *robots are agents*, who should be capable of execution-time reasoning and management of their interactions with their peers. Such reasoning alleviates reliance on being able to account for every possible interaction in advance. To manage these interactions in execution time, this paper presents BIS (Bar Ilan Single), a general plan-execution algorithm. The design of BIS neatly separates knowledge (beliefs) and decision-making procedures (plans).

Keywords: Distributed multi-robot systems, collaboration, teamwork

1. The Individual Robot

We begin by considering an abstract modern robot, capable of making decisions during execution-time. Such a robot is necessarily equipped with computational processes for *decision making*, and for perception (more generally, *world modeling*), which provides the basis for making decisions. Perceptual and world modeling processes turn sensor readings and task parameters into beliefs, and maintain those over time. SLAM (Simultaneous Localization and Mapping) is a great example of such process. Vision-based segmentation is another. Decision making processes deal with turning goals and beliefs into actions: given the goals of the robot, and its beliefs as to current state of the world, decision-making processes decide on the next action to take, e.g., by selecting such actions from relevant plans.

1.1. Beliefs

I treat knowledge abstractly here. A belief (a piece of knowledge) k is a tuple $\langle n, v \rangle$, where n is a unique variable name, and v its value. The variable name is a pointer to the storage in which the value can be stored. v can be as simple as a number or string, or as a complex as a structured graph, or a procedure. For instance, the tuple $\langle$laser_readings, [...]$\rangle$ may refer to a vector containing the current laser range readings, index by the angle. Or the tuple $\langle$predicate_testing_distance_over_10, *true*$\rangle$ contains the result of testing whether the value of another piece of knowledge, containing distance to an obstacle, is greater than 10.

I fully acknowledge that this is a very simplistic way of looking at knowledge. There is a practically endless supply of previous works who have investigated how knowledge can be stored, retrieved, represented, inferred, merged, revised, synthesized (e.g., a pose estimation carried out by a localization process, based on sensor readings), learned, and manipulated. There is no way to do justice to these in a single book, let alone this single paper. Fortunately, there is no need. I utilize this simplistic view of knowledge because for the purposes of this paper (1) I do not need to discuss inference, reasoning or monotonicity, and (2) treating knowledge as variable-value pairs keeps the discussion close to the implementation. In other words, this paper ignores the inference of knowledge, and can therefore ignore the full extent of knowledge representation.

Email address: galk@cs.biu.ac.il (Gal A. Kaminka)

1.2. Decisions

Decisions are made by robots, during execution, when their plans are not policies, i.e., when their plans do not fully dictate their action at all times. Instead, the plans that the robots have serve to only guide and constrain their reasoning; they provide possible recipes for action, which the robot must consider against its own state and goals, the environment state, and the state of its peers. Decisions are made to follow, modify, or ignore a plan. Therefore, before discussing decision-making, it is necessary to first describe plans.

The basic unit of a plan is a *behavior* [5, 1]). A behavior is a controller coupling perceptions (inputs from the world modeling and perception component) and actions (outputs to the actuators and motor controllers). It may make use of memory by writing to the world modeling component,i.e., forming new beliefs or revising old ones. A behavior carries out a sequence of actions intended to bring about a limited goal (e.g., kick a ball in a given direction), or maintain one (e.g., maintain range and bearing with respect to another robot). Behaviors are assumed to be executable by the robot; the question is how to sequence their execution.

Borrowing from BDI (Belief Desire Intention) architectures (e.g., [4, 2, 6]), behaviors are collected together to form *plans*. This is done in two ways. First, behaviors are arranged sequentially, describing possible sequences of plan-steps (each a behavior). Second, plans are also described hierarchically, so that plans can be composed of sub-plans. Note that both sequentially and hierarchically, there is no commitment in a plan to only one combination or one ordering of behaviors. Rather, edges denote possibilities, not commitments.

Formally, a plan is an augmented connected directed graph, defined by a tuple $\langle B, H, N, b_0 \rangle$, where B is a set of vertices representing behaviors, H and N are sets of directed edges ($H \cap N = \emptyset$), and $b_0 \in B$ is a behavior in which execution begins. N is a set of *sequential* edges, which specify temporal order of execution of behaviors. Given $b_1, b_2 \in B$, a sequential edge from b_1 to b_2 specifies that b_1 must be executed before executing b_2. A path along sequential edges, i.e., a valid sequence of behaviors, is called an *execution chain*. H is a set of hierarchical *task-decomposition* edges, which allow a single higher-level behavior to be broken down into execution chains containing multiple lower-level behaviors. As a matter of notation, when picturing graphs in figures, we will show edges in H only when they point to the first behavior in an execution chain. Sequential edges may form circles (to indicate that a behavior may repeat itself), but decomposition edges cannot. Thus behaviors can be iterated by choice, but cannot be their own ancestors.

The incompleteness of plans—the difference between plans and policies—comes into play in that in general, the plan does not, in advance, know which exact sequence or hierarchical decomposition of behaviors will achieve its goal, nor the duration of each behavior's execution. Instead, each plan-step behavior $b \in B$ has a set of preconditions (denoted preconds(b)) which—when satisfied—enable its selection (the robot selects between enabled behaviors), and termination conditions (denoted termconds(b)) that determine when its execution must be stopped (note these are not predicted effects, as in planning operators). Preconditions and termination conditions test perceptions and the internal state variables of the robot. In this, behaviors used in these plans are similar to *options*, in reinforcement learning [7].

When the robot starts (by selecting the behavior b_0 for execution), it first engages in hierarchical decomposition, until a complete hierarchical path through the behavior graph—b_0 through hierarchical edges, to an atomic behavior b_a is selected. During this process, the preconditions of sub-plans (which begin execution chains) are matched against the world model, to determine whether sub-plans are selectable. The robot chooses among alternative decompositions (see below). Execution of the selected behaviors then commences. The termination conditions of all running behaviors are continuously matched against the world model, which is itself continuously updated by the perceptual processes, and optionally by the behaviors themselves writing to internal state variables). Once one or more of the behaviors signals it is ready for termination, its execution stops, as well as that of its running children. The robot then chooses from enabled behaviors next in the sequence (if any), or goes back to the parent behavior which is still running. This process—for the individual robot—is described is Algorithm 1.

In general, the robot must deliberate. It must make decisions at run-time, between alternative execution chains, and between alternative hierarchical decompositions. There is no one general method for doing this: One could apply decision-theoretic reasoning, look at activation functions that match the state, apply limited simulation to evaluate the potential success of each behavior, use heuristic search through future selections, etc. The algorithm takes a neutral approach. It delegates the decision to the CHOOSE procedure, whose task is to apply a decision procedure provided by the planner, to the decision at hand.

This mechanism does not shirk the responsibility of making a selection. Rather, it recognizes that just as the commitment to the selected behavior should be made as late as possible, so does the commitment to the decision

mechanism used to make this choice. This lazy-commitment strategy is one primary difference between the plan representation presented here, and, say, a finite state machine, where commitments are made early, during planning time.

Algorithm 1 Individual decision-making algorithm.

Require: Plan $P = \langle B, H, N, b_0 \rangle$
Require: Knowledebase W
Require: Choice Procedure CHOOSE
Require: Condition Testing Procedure TEST
Require: Belief Update Procedure UPDATE
Require: Belief Revision Procedure REVISE
Require: Start Execution Procedure START
Require: Stop Execution Procedure STOP

1: $S \leftarrow \emptyset$ ▷ New execution stack
2: $b \leftarrow b_0$
3: PUSH(S,b)
4: **while** $\exists n$, where $(b, n) \in H$ **do** ▷ b can be decomposed
5: $A \leftarrow \{n | (b, n) \in H\}$ ▷ children of b
6: $C \leftarrow \{a | a \in A, \text{TEST}(\text{preconds}(a), W)\}$
7: $b \leftarrow \text{CHOOSE}(C, P, S, W)$ ▷ Choose among C behaviors
8: PUSH(S,b)

9: **for all** $s \in S$ **do** ▷ Start execution of all behaviors in S
10: **if** s **not running** START(s)

11: $E \leftarrow \emptyset$
12: **while** $E = \emptyset$ **do** ▷ E is the set of terminating behaviors
13: $K \leftarrow \text{UPDATE}(W)$
14: $W \leftarrow \text{REVISE}(W, K)$
15: $E \leftarrow \{a | a \in S, \text{TEST}(\text{termconds}(a), W)\}$ ▷ Check termination conditions

16: **while** $E \neq \emptyset$ **do** ▷ Stop and pop all terminating behaviors and descendants
17: $e \leftarrow Pop(S)$
18: STOP(e)
19: **if** $e \in E$ **then**
20: $E \leftarrow E - \{e\}$

21: $A \leftarrow \{n | (e, n) \in N\}$ ▷ sequential followers of e, topmost terminating behavior
22: $C \leftarrow \{a | a \in A, \text{TEST}(\text{preconds}(a), W)\}$
23: **if** $C \neq \emptyset$ **then** ▷ There are potential followers
24: $b \leftarrow \text{CHOOSE}(C, P, S, W)$ ▷ Choose among C behaviors
25: **Goto** 3
26: $b \leftarrow \text{PEEK}(S)$ ▷ No potential followers, continue with parent
27: **if** $b \neq \emptyset$ **then**
28: **Goto** 11
29: **Halt**.

Besides the CHOOSE procedure, Algorithm 1 also relies on procedures for accessing a stack which holds the behaviors selected for execution, for getting new updates from the world modeling processes (UPDATE), and for merging these updates with the knowledge-base that the robot maintains of its beliefs (REVISE). The procedures START and STOP are used to begin and end execution of a given behavior. In that, I am adopting a view of behaviors as separate threads, which are controlled from Algorithm 1.

[1] R. C. Arkin. *Behavior-Based Robotics*. MIT Press, 1998.
[2] M. J. Huber. JAM: A BDI-theoretic mobile agent architecture. In *Proceedings of the Third International Conference on Autonomous Agents (Agents-99)*, pages 236–243, 1999.
[3] G. A. Kaminka, A. Yakir, D. Erusalimchik, and N. Cohen-Nov. Towards collaborative task and team maintenance. In *Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-07)*, 2007.
[4] J. Lee, M. J. Huber, P. G. Kenny, and E. H. Durfee. UM-PRS: An implementation of the procedural reasoning system for multirobot applications. In *Proceedings of the Conference on Intelligent Robotics in Field, Factory, Service, and Space (CIRFFSS-94)*, pages 842–849, 1994.
[5] R. R. Murphy. *Introduction to AI Robotics*. MIT Press, 2000.
[6] A. Newell. *Unified Theories of Cognition*. Harvard University Press, Cambridge, Massachusetts, 1990.
[7] R. S. Sutton, D. Precup, and S. P. Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1–2):181–211, 1999.