Knowledge Considerations in Robotics and Distribution of Robotic Tasks

Ronen I. Brafman and Yoav Shoham
Robotics Laboratory
Stanford University
Stanford, CA. 94305
{brafman,shoham}@cs.stanford.edu

Abstract

We develop a formal tool for representing and analyzing informational aspects of robotic tasks, based on the formal concept of 'knowledge.' Specifically, we adopt the notion of knowledge-based protocols from distributed systems, and define the notions of knowledge complexity of a robotic task and knowledge capability of a robot. The resulting formalism naturally captures previous work in the areas of robot information management, but is sufficiently rigorous and natural to allow many extensions. In this paper we show one novel application - the automated distribution of robotic tasks.

1 Introduction

The notion of computational complexity has had a profound effect on the development of computer science. While still crude, our ability to classify different computational problems in terms of their complexity allows us to understand inherent difficulties in solving such problems. Some areas of robotics, such as motion planning, have benefited from advances in computational complexity (e.g.,[Canny, 1989]). However, the area of robotics as a whole still lacks the analog of a Turing machine, a formal device that faithfully quantifies the difficulty of a robotic task or the capabilities of a robot. The reason for this is that usually, space and time complexity of computation are not the dominating factors in a robotic task. Instead, the physical embedding of robots brings to the fore issues such as sloppy controllers, imprecise sensors, and spatially separated components, all of which suggest that a good model for robotics should revolve around the notions of information and uncertainty.

We propose a framework that faithfully models the information and lack thereof inherent in robotic tasks, such as is brought about by spatial distribution and imprecise sensors. In doing so we join a number of authors who have recently attempted to quantify the sensing difficulty of robotic tasks. [Erdmann, 1994] attempts to assess the sensing requirements of robotics tasks in terms of what he calls abstract sensors. [Donald, 1994] develops formal tools that allow him, among other things, to compare and classify the sensing capabilities of different concrete sensor systems such as a radial sensor and a beacon sensor. We find this recent work inspiring, but believe that further progress can be made by couching the analysis of informational considerations in robotic tasks in a suitable abstract language. In particular, we believe that the formal notion of 'knowledge' that has been proposed and studied in AI (e.g., [Moore, 1985; Rosenschein, 1985]) and distributed systems (e.g., [Fagin *et al*., 1995]) can serve as a basis for just such an abstraction. We recently showed a particular application of the formal notion of knowledge to the task of robot motion planning under uncertainty [Brafman *et a*/., 1994]; here we propose a more ambitious application of the notion, namely as a basis for the general model of informational aspects of robots and robotic tasks. In our framework, robotic tasks can be characterized in terms of the knowledge required to perform them, and robots can be characterized in terms of the knowledge they can acquire. We can therefore assess the ability of a particular robot to perform a task by comparing its knowledge capabilities to the knowledge requirements of the task. One of the benefits of this formal approach and its underlying semantics is extensibility. In particular, we are able to exploit this feature in the analysis of multi-robot domains. As a demonstration of this, we provide a provably correct algorithm for distributing centralized robotic protocols; roughly speaking, the algorithm accepts as input a description of a robotic task and a high-level description of a centralized protocol for achieving the task, and outputs a high-level description of a decentralized protocol that is guaranteed to achieve the same task.

To provide intuition, throughout the paper we will anchor the formal development in the following example. Although simple, the example embodies two important ingredients - imprecise sensing, and the need to coordinate the action of spatially distributed actuators.

Example 1. Two horizontal, perpendicular, one-dimensional robotic arms must coordinate as follows. The first arm is to push a hot object lengthwise across the table until the the second arm is able to push it sideways so that it falls into a cooling bin. The length of the table is marked in feet, from 0 thru 10 (for simplicity we ignore the horizontal coordinate). The second arm is able to push the object if it is anywhere in the region [3,7]. The second arm cannot hit the object while the object is in motion, but on the other hand the object cannot remain motionless for more than an instant

or it will burn a hole into the table. Thus the second arm must move precisely when the first one stops. We consider four variants of the problem:

Ia. The arms share a controller. The controller has access to a sensor reporting the position of the object with error no greater than 1, i.e., if the object's current location is q then the reading can be anywhere in $[q - 1, q + 1]$.

Ib. Same as Ia, except the error bound is 4 rather than 1.

Ic. Each arm is controlled separately. Each controller has access to a location sensor as in Ia; however these are two independent sensors whose readings may differ (within the allowed bounds).

Id. The situation is as in Ic, except that only the first controller has a sensor. However, in addition the first controller can emit (e.g., infra-red) signals, and the second controller can reliably detect them.

It is not hard to see that in cases Ib and Ic there does not exist a protocol that will achieve the task, whereas in cases Ia and Id there do exist such protocols.[1] |

There are three remaining sections in the article: (2) the formal model and language, (3) an application to the distribution of robotic tasks, and *(A)* discussion of related work.

2 The model

We start by presenting our model, which is based on the notions of *Abstract Robotic Domain* (ARD) and *Abstract Robotic Unit* (ARU). In the second section we present a (by now standard) logic for talking about the model, including the knowledge of ARUs. In the third section we introduce the new notions of *knowledge capability* and *knowledge complexity,* which we then relate to the notion of *knowledge-based protocols.*

2.1 Abstract Robotic Units and Domains

We start by defining the notion of an *Abstract Robotic Domain,* or *ARD.* An ARD consists of a set of possible states and the possible transformations among them. Roughly, a state of the ARD corresponds to the notion of *configuration* in robotics [Lozano-Perez, 1983]; it encodes everything that is not internal to robots, such as the location of the robot and other objects. The information and computation capabilities of a robot are captured by an *Abstract Robotic Unit (ARU).* An ARU is a state machine whose actions "cause" the ARD's transitions. Different ARUs acting on a given ARD have identical actuation capabilities, but their abilities to sense and compute can vary.

Definition 1 *An ARD is a pair (C,T) where C, the* configuration space, is *a set of states, and T, the* transitions, *is a set of functions from C to* $2^{C}\backslash 0$, *containing the identity function.*

An ARU *over an ARD* $(\mathcal{C}, \mathcal{T})$ *is a triple* $(States, Actions, Init)$, *where States is the set of* local states *of the ARU, Actions is a set of functions from* $GS = \mathcal{C} \times States$ *to* $2^{GS}\backslash\emptyset$, *and Init is a function from* $\mathcal{C}$ *to* $2^{States}\backslash\emptyset$. *GS is called the set of* global states. *We say that* $f \in Actions$ implements *the transition* $\tau \in \mathcal{T}$ *if for all* $c, c' \in \mathcal{C}$ *and* $state, state' \in States$ *it is the case that if* $(c', state') \in f(c, state)$ *then (1)* $c' \in \tau(c)$; *(2) if* $c'' \in \tau(c)$ *then there exists some* $state'' \in States$ *such that* $(c'', state'') \in f(c, state)$. *The actions Actions of an ARU over ARD* $(\mathcal{C}, \mathcal{T})$ *must all be implementations of transitions in* $\mathcal{T}$, *and must implement all such transitions.*

Given a set of initial configurations $I \subset \mathcal{C}$ *the set of* reachable (global) states $S[I]$ *for an ARU* $(States, Actions, Init)$ *is the smallest subset of GS that contains* $\{(c, l) | c \in I \text{ and } l \in Init(c)\}$ *and is closed under Actions.*

The projections of a global state $(c, l) \in GS$ *are defined as* $\pi_{config}(c, l) \stackrel{\text{def}}{=} c$ *and* $\pi_{local}(c, l) \stackrel{\text{def}}{=} l$.

We will sometimes refer to an ARU as a 'robot,' but there is no requirement that the sensors and effectors make for a contiguous piece of equipment or that they are otherwise related to one another. For this reason it is often convenient to view an ARU as comprising of $\text{ARU}_1, \ldots, \text{ARU}_n$, where $\text{ARU}_i = (States_i, Actions_i)$ has its own local states and actions. We refer to this special type of ARU as *n-ARU* and to its subcomponents as *sub-ARUs.* The global states of the n-ARU are thus $\mathcal{C} \times States_1 \times \cdots \times States_n$, and its actions are the joint-actions of the sub-ARUs, i.e., $Actions_1 \times \cdots \times Actions_n$.

Throughout this paper we assume a discrete model; specifically, we assume all state spaces to be discrete, and adopt a discrete model of time. We do so to simplify the exposition; the generalization to continuous domains is for the most part straightforward, though some subtleties do arise (cf.[Brafman *et al.*, 1994] for an application of logics of knowledge in motion planning in domains that are continuous in both time and space).

Example 1 (cont.) Here are an ARD and an ARU that model the controller Ia:

$\text{ARD}_{1a} = ([0, 10] \times \{Table, Side\}, \{\text{Move-length}, \text{Move-side}\})$ where Move-length transforms $(q, Table)$ to $(q + 1, Table)$, with no effect otherwise or when $q > 9$. Move-side transforms $(q, Table)$ to $(q, Side)$, and does nothing elsewhere. The initial configuration is (0,Table).

$\text{ARU}_{1a} = (\{[0,10], \{\text{Move}(\text{arm}_1), \text{Stop}(\text{arm}_1)\} \times \{\text{Move}(\text{arm}_2), \text{Stop}(\text{arm}_2)\} \times \text{Sense-position}\}, I_{1a})$, where $q' \in I_{1a}(q)$ iff $|q' - q| \leq 1$. That is, the possible configurations correspond to positions of the hot object, transitions correspond to length-wise or sidewise motions, and initial local states and positions differ by at most 1. The controller's local state consists of a position reading, and its actions are joint motions of both arms, combined with sensing. An example of an action description would be: $(\text{Move}(\text{arm}_1), \text{Stop}(\text{arm}_2), \text{Sense-position})((q, Table), r) = \{((q', Table), r') | (q', Table) = \text{Move-length}(q, Table) \ \& \ |q' - r'| \leq 1\}$. This description is consistent with the fact that the position and position reading of reachable global states can differ by at most 1. ∎

From now on we fix the physical characteristics of the environment, i.e., we assume a given, fixed ARD $(\mathcal{C}, \mathcal{T})$; this allows us to concentrate on epistemic considerations.

1 For example, the centralized protocol in case Ia might be as follows (r is the current reading): If r < 4 then (Move(armi),Stop(arm2)); else (Stop(armi),Move(arm2)). Similarly, a decentralized protocol in case Id might be:
Protocol for arm_1: If r < 4 then Move; If r E [4,6] then signal and stop.
Protocol for arm_2: If Signal has been detected then Move; Else stop.

Next, we define a number of straightforward notions. A *history* over a set W is simply a sequence of elements of W. We define a *task* to be a pair of sets of configurations, which intuitively correspond to the set of initial and goal configurations. (A task can be defined more generally as an arbitrary set of histories over $\mathcal{C}$, but in this paper we avoid this generality.) We say that an ARU *can perform* a task if it can be equipped with a controller that will lead it to a goal state from any initial configuration. To describe controllers we use the notion of a *protocol* (or policy), an assignment of actions to local states. Each protocol has a set of possible *executions*, i.e., sequences of global states that can be obtained when this protocol is followed.

Definition 2 *A* history *over a set W is a function $h : \mathcal{N} \to W$, where $\mathcal{N}$ are the integers.*
A task *is a pair $Task(I, G)$, where $I, G \subset \mathcal{C}$.*
A protocol *for an ARU is a function $\mathcal{P}$: States $\to$ Actions.*
A distributed protocol *for n-ARU is a set of functions $\mathcal{P}_i$: States$_i$ $\to$ Actions$_i$ for $i = 1, \ldots, n$.*
A joint-protocol *is a (centralized) protocol for an n-ARU, i.e., a function from States$_1$ $\times \ldots \times$ States$_n$ to Actions$_1$ $\times \ldots \times$ Actions$_n$.*
An execution *of the protocol $\mathcal{P}$ is a history h over GS such that $h(n+1) \in \mathcal{P}(\pi_{local}(h(n)))(h(n))$.*
A protocol $\mathcal{P}$ performs *$Task(I, G)$ if every execution of $\mathcal{P}$ that starts in $\{(c, l) | c \in I$ and $l \in Init(c)\}$ passes through a configuration in G.*
An ARU can perform *$Task(I, G)$ if there is a protocol for it that performs $Task(I, G)$.*

2.2 Talking about what ARUs know

The model given in the previous section provides a stable basis, but the real power will come from the language with which we choose to speak about the model. The following definitions are by now standard in the formal AI and distributed computing communities (cf. [Rosenschein, 1985] and [Halpern and Moses, 1990]), but are probably new to many robotics researchers. While we include all definitions here, we will be able to provide full intuition for those new to the area only in a long version of this paper.

Assume a robotic system with a set $\mathcal{S}$ of reachable global states (i.e., configuration states as well as local states). To be able to talk about the properties of states in $\mathcal{S}$ we introduce a propositional language $\mathcal{L}$, constructed from a collection P of primitive propositional symbols and the boolean connectives $\neg$ and $\wedge$ (we restrict the article to the propositional case). This language describes properties of the system's configuration, such as the robot's being in the goal. We then add an interpretation function π that assigns a truth value to elements in P. Given $\mathcal{L}$ and π we define the satisfiability of a formula $\varphi \in \mathcal{L}$ in a state $s \in \mathcal{S}$, written $\mathcal{S}, s \models \varphi$, as follows:

- $\mathcal{S}, s \models p$ for $p \in P$ iff $\pi(s, p) = true$;
- $\mathcal{S}, s \models \neg\varphi$ iff it is not the case that $\mathcal{S}, s \models \varphi$;
- $\mathcal{S}, s \models \varphi \wedge \psi$ iff $\mathcal{S}, s \models \varphi$ and $\mathcal{S}, s \models \psi$;
- $\mathcal{S} \models \varphi$ iff for all $s \in \mathcal{S}$ we have that $\mathcal{S}, s \models \varphi$.

We now augment the language with the notion of knowledge. We say that an ARU in local state l *knows* a wff φ if φ holds in all global states in which the ARU's local state is l. Formally:

- $\mathcal{S}, s \models K_a\varphi$ iff $\mathcal{S}, s' \models \varphi$ for all $s' \in \mathcal{S}$ such that sub-ARU a's local state is identical in s and s'.[2]

Example 1 (cont.) Consider variant 1a, and let the proposition g stand for the assertion "the object is in the goal region, i.e., in the region [3,7]." When the controller has the position reading of 4, it might in fact be in one of three actual positions: 3,4 and 5. However, since all these positions satisfy g, we can say that it *knows* g at this state. ∎

As is well known in AI and distributed systems, the properties of this definition can be captured in a small collection of axioms known as the *S5 axiom system*. Going into the properties of S5 is beyond the scope of the paper, but we do remark that these properties bear only a crude similarity to the commonsense notion of knowledge. However, this less-than-perfect match will not make the formal notion any less useful for our purposes.

2.3 Knowledge complexity

Having defined an implementation-independent notion of knowledge, we are in a position to define a complexity measure over robotics tasks and a capability measure for robots. We first define the capability of a robot to attain knowledge. Intuitively, for a robot to know something it must either already possess the knowledge, or else be able to take actions at the conclusion of which the knowledge is guaranteed. These actions may be simple sensing, running an internal routine, or navigating around the environment to gather information. In this article we allow only 'non-destructive' knowledge acquisition; that is, we allow this to happen through actions that culminate in additional knowledge (and thus change in local state) to the robot, but in no change to the system's configuration.

Definition 3 *An ARU is* K-capable *of $\{\varphi_0, \ldots, \varphi_k\} \subset \mathcal{L}$ in a set of global states S if there exists a protocol $\mathcal{P}$ such that for every $s \in S$, every execution of $\mathcal{P}$ that starts at s reaches a state s' such that (a) $\mathcal{S}, s' \models K\varphi_0 \vee \cdots \vee K\varphi_k$, (b) s and s' differ at most in the local state of the ARU, and (c) the ARU knows that (b) holds.*[3]

The knowledge-capability of a robot is characterized by a set of propositions, the interpretation of which is that the robot can come to know at least one of those at each configuration. This definition embodies the notion that sensing is nondeterministic. The robot may be able to guarantee that it come to know one of several facts,

[2] We use a subscripted K to distinguish between the knowledge of different sub-ARUs. This will also allow us to make statements such as 'sub-ARU a knows that sub-ARU b knows φ': $K_aK_b\varphi$. When the the ARU is obvious from the context we will drop the subscript, as in $K\varphi$.

[3] This is a subtle point which will be discussed in a longer version of this paper. We briefly mention that (b) and (c) always hold when we restrict $\mathcal{P}$ to purely sensory actions, i.e., actions that never change the configuration. Alternatively, (c) can be handled by crafting a class of protocols with an explicit termination condition.

but not any one of them in particular. For example, given a position sensor with ±1 error, sensing the position when in location 4 will yield knowledge of one of the following three facts: "the location is in the 2-4 region," "the location is in the 3-5 region," and "the location is in the 4-6 region." Yet, knowledge of any one particular statement is not guaranteed.

Definition 4 *$\{\varphi_0,\ldots,\varphi_k\}$ is an* upper bound *on the K-complexity of a task $Task(I,G)$, written $Task(I,G) = O(\{\varphi_0,\ldots,\varphi_k\})$, if (1) any ARU that is K-capable of $\{\varphi_0,\ldots,\varphi_k\}$ in $\mathcal{S}[I]$ can perform $Task(I,G)$; (2) Some ARU is K-capable of $\{\varphi_0,\ldots,\varphi_k\}$ in $\mathcal{S}[I]$.*

The notion of a lower bound can be defined similarly, but it does not play a role in the sequel.

We make two observations about K-complexity. First, it differs from the notions of time and space complexity in that K-complexity values are not totally ordered. Second, there appears to be an interaction between K-complexity on the one hand and time and space complexity on the other. Intuitively, while a robot with minimal knowledge might be able to perform a task, with more knowledge it might be able to perform the task more efficiently in terms of computation time or space. We do not explore this interesting tradeoff here, but we believe it represents an important issue.

Example 1 (cont.) Consider variants 1a and 1b. Recall that in both variants a central controller is in charge of a two-armed robot, which must perform two actions simultaneously when the object is in the goal region $G = \{3,4,5,6,7\}$.

K-complexity: It is easy to see that this task is $O(\{g,\neg g\})$. However, this is not a tight bound. If in positions 3 or 4 the robot is not sure where it is, it can still manage, as long as it is able to detect the goal later. We can therefore tighten the boundto $O(\{g,\varphi\})$, where φ denotes being in $\{1,2,3,4,6,7,8,9,10\}$. That is, in $G \setminus \{1,2,3,4,6,7,8,9,10\}$ the robot must know that it is in the goal; in $\{1,2,3,4,6,7,8,9,10\} \setminus G$ it must know φ, and in $G \cap \{1,2,3,4,6,7,8,9,10\}$ it should know either one.

K-capability in 1a: The first controller's sensor has an error bound of 1. To verify that it is K-capable of $\{g,\varphi\}$ we must see that it always satisfies $Kg \vee K\varphi$. This is easily verified. For example, in position 3, its possible readings are 2,3,4, each of which makes it know g or φ. We conclude that this controller can perform this task.

K-capability in 1b: The second controller's sensor has an error bound of 4. Given this error bound it cannot satisfy Kg at any state. g, or a stronger formula, must appear in any K-complexity bound of this task, since any protocol that allows Move(arm_2) at a point that does not satisfy g will fail. We conclude that this controller cannot perform the task. ∎

2.4 Knowledge-based protocols

Work in distributed systems has demonstrated that the formal notion of knowledge is a powerful tool for analyzing traditional protocols. However, it was recently noted that the notion could be useful for design purposes too; it allows one to design high-level protocols that concentrate on the informational aspects of the task without drowning in implementation details. We define a slight variant of an abstract notion of protocol - *knowledge-based protocols* (KBP), due to [Fagin *et al.*, 1995].[4] A KBP is a (possibly partial) function from the set of knowledge-states to the transitions of the given ARD. It can be viewed as a big case statement, each condition of which is a test on the knowledge of the ARU. The interpretation of this protocol is that the ARU always performs an action that implements the transition corresponding to the *earliest* condition satisfied. A KBP can also be represented in disjoint form, in which this priority is made explicit. Here are these two forms:

KB-protocol		KB-protocol in disjoint form:	
Case:		Case:	
$K\alpha_1$	τ_1;	$K\alpha_1$	τ_1;
$K\alpha_2$	τ_2;	$K\alpha_2 \wedge \neg K\alpha_1$	τ_2;
...		...	
$K\alpha_n$	τ_n;	$K\alpha_n \wedge \bigwedge_{1\le i<n} \neg K\alpha_i$	τ_n;

We refer to $K\alpha_1,\ldots,K\alpha_n$ as the (knowledge) *conditions* of this KBP. We say that an ARU *can run* a KBP with conditions $K\alpha_1,\ldots,K\alpha_n$ in $\mathcal{S}$, if $\mathcal{S} \models \bigvee_{i=1,\ldots,n} K\alpha_i$. Thus, a KBP specifies what to do and a class of ARUs that are capable of following this specification. This last point is an important difference between our definition and that of [Fagin *et al.*, 1995]. Their KBPs contain a final ELSE clause, and are thus executable by any ARU.

In principle, there is always a well-defined translation from a standard protocol to an equivalent KBP. A standard protocol can be viewed as a special case of a KBP, since, by definition, an ARU always knows its local state. Conversely, a *standard translation* of a KBP $\mathcal{P}$ for an ARU and a set of global states $\mathcal{S}$ is obtained as follows: Let $K\varphi$ be the earliest condition in $\mathcal{P}$ that is satisfied in the local state l, and let τ be the corresponding transition in $\mathcal{P}$. A standard translation assigns to l an action that implements τ. This translation results in a complete protocol (i.e., one in which an action is assigned to every state) when the ARU can run the KBP in $\mathcal{S}$. In the sequel we are only interested in KBPs that can be run by some ARU in an appropriate set of global states.

Informally, we say that h is an *execution* of a KBP if it is an execution of some standard translation of it. Formally, the executions of a KBP $\mathcal{P}$ from initial configurations $I \subset \mathcal{C}$ are those executions h of a standard protocol $\mathcal{P}_s$ by an ARU such that: (1) $h(0) \in \{(c,l) | c \in I \text{ and } l \in Init(c)\}$, (2) the ARU can run $\mathcal{P}$ in $\mathcal{S}[I]$, (3) $\mathcal{P}_s$ is a standard translation of $\mathcal{P}$ for this ARU and $\mathcal{S}[I]$. We say that KBP $\mathcal{P}$ *performs* $Task(I,G)$ if all its executions from initial configurations in I pass through a configuration in G, and if the set of executions from I is not empty.

Notice that an assignment of the identity transition by a KBP is redundant (except in goal states) since it does not change the configuration. We therefore ignore such KBPs in the sequel.

The following lemma requires an additional definition: φ is *about* $\mathcal{C}$ if for all $s,s' \in \mathcal{S}$: if $\pi_{config}(s) = \pi_{config}(s')$ then $\mathcal{S},s \models \varphi \Leftrightarrow \mathcal{S},s' \models \varphi$.

Lemma 1 *Let $\varphi_1,\ldots,\varphi_k$ be about $\mathcal{C}$ in $\mathcal{S}[I]$ for any ARU. $Task(I,G) = O(\{\varphi_1,\ldots,\varphi_k\})$ iff there is*

[4] The similarity in names between KBPs and knowledge-based (or expert) systems in AI is coincidental.

a KBP with conditions $K\varphi_1, \ldots, K\varphi_k$ that perforins Task(I,G).

This lemma suggests a method for analyzing a robot's ability to perform a task in terms of two subproblems: (1) finding a KBP *V* that performs the task, and (2) showing that the robot is K-capable of $\{\varphi_1, \ldots, \varphi_n\}$, where $K\varphi_1, \ldots, K\varphi_n$ are the knowledge conditions of *P*. This robot can then perform the task by running a KBP derived from *V* by adding a final else clause which (abstractly) states:

Else learn whether φ_1 or ... or φ_n

Thus, the robot will alternately be making direct "physical" progress toward the goal state and attaining the knowledge required to make such progress.

We conclude the section with two remarks on KBPs. First, although, as we have just mentioned, a standard protocol can always be viewed as a special case of a KBP, it is a degenerate case. KBPs are powerful precisely because they allow us to abstract away the idiosyncrasies of local state. Thus, for example, rather than discuss the content of the frame buffer of a robot's vision system, a KBP allows us to talk about the robot knowing that there is an obstacle in front of it. Second, while in principle well defined, in practice it can be quite difficult to transform a KBP to a standard one. We believe, however, that, as has been the case in distributed systems, starting with knowledge-oriented analysis and design is a useful methodology.

Example 1 (cont.) Let *g* and alpha be as defined before. Here is an example of a distributed KBP for the task, which we derive algorithmically in Section 3:

Protocol for arm$_1$*.if* $K\backslash g$ then Stop; if $K_1\varphi$ then Move.

Protocol for arm K_2K_1g then Move; if $K_2(K_1\varphi \wedge \neg K_1g)$ then Stop.

As we will see, the use of nested knowledge operators is a natural and concise way of capturing coordination. |

3 Automated distribution of robotic tasks

So far we have provided a rigorous framework for representing and reasoning about informational aspects of robots and robotic tasks. Certainly, a minimal requirement from any formal framework is that one be able to use it to reason about simple and intuitively well understood examples; indeed, all along we have applied the framework to such an example. However, for the framework to be something other than an idle exercise, it must ultimately be used for other than merely formalizing the obvious. One direction to go would be to increase the complexity of the robotic task until its solution or lack thereof are no longer obvious. However, here we offer a different novel contribution - a provably correct algorithm for distributing robotic tasks; that is, an algorithm that (roughly speaking) accepts a multi-robot task and a central controller for the robots that achieves it, and outputs local controllers for each of the robots that jointly achieve the same task.

Our approach to task distribution rests on capturing the notion of centralization through the formal notion of knowledge. With central control all knowledge of the various components resides at one place, the controller. This has far-reaching consequences. For example, in a two-armed robot, a central controller means that not only do both arms always have the same knowledge, but they each know that they do, they each know that they each know, and so on; this is called *common knowledge* in the logic literature. Endowing the two arms with separate controllers breaks this common knowledge; the arms may have different knowledge, or may have the same knowledge without knowing that they do. Our knowledge-level language allows us to identify the levels of knowledge required of different parts of the system, and thus to precisely quantify various degrees of centralization.

Recall that our formal language $\mathcal{L}$ allows nesting of the knowledge operator. For example, the sentence $K_1K_2\varphi$ reads "ARU 1 knows that ARU 2 knows that φ is the case." Recalling that "knowledge" is merely a way of capturing a certain correlation between an ARU's local state and the global state of the system, this means that the above statement concisely and elegantly captures a complicated relationship between the local state of ARU 1, the local state of ARU 2, and the system's configuration. Knowledge operators can be nested more deeply, encoding even more complex relationships between states; $K_2K_1K_2\varphi$ is perfectly legal.

Before describing our algorithm we must explain the notion of a *progress measure,* which originates in the program verification and synthesis literature. The general idea is to assign values to states from some well-ordered set (i.e., a set whose every subset has a minimal element) with the goal state assigned the minimal value. Given a progress measure we can prove that a protocol performs a task by showing that each transition the protocol takes produces a new state whose value is smaller than that of the previous state. [Erdmann, 1994] ingeniously employs these ideas to the problem of generating minimal sensors. He uses an existing protocol to generate a progress measure that assigns to each state the maximal number of steps that the protocol might take to achieve the goal. He then uses this progress measure to obtain sufficient sensing requirements for a task. We use his method as a "subroutine" in the algorithm.

Definition 5 *A* progress measure *for an ARD* (C, T) *is a functi* $\rho(\mathcal{C}) \longrightarrow W$, *where W is a well-ordered set.*

We can construct a progress measure given a protocol and a task. The idea is to see for each state what is the longest the protocol might take to lead the system to a goal state.

Definition 6 *Given a task Task*$\backslash(I, G)$ *and a KBP P, the f u n* $\rho_{(\mathcal{P},Task(I,G))}$*s defined for each* $c \in \mathcal{C}$ *as the least upper bound on the number of steps along any execution of V from initial configuration {c} until the first time a configuration in G is reached, or* ∞ *if no such upper bound exists or if a configuration is reached on which V is undefined. We say that* $\tau \in \mathcal{T}$ makes progress *on* $c \in \mathcal{C}$ *according to p if for any* $c' \in \mathcal{T}(c)(c)$, $\rho(c') < \rho(c)$

Lemma 2 $\rho_{(\mathcal{P},Task(I,G))}$ is a *progress measure.*

We will also need the following:

Definition 7 *Given a joint KBP $\mathcal{P}$ for an n-ARU, the i^{th} projection of $\mathcal{P}$ is a KBP for ARU_i containing a pair of the form* "if $K_i\varphi$ then do τ_i" *whenever $\mathcal{P}$ contains* "if $K\varphi$ then do $(\tau_1, \ldots, \tau_i, \ldots, \tau_N)$".

A configuration c possibly satisfies *φ if there is an ARU, a set of global states $\mathcal{S} \subset GS$, and some $s \in \mathcal{S}$ such that $S, s \models \varphi$ and $\pi_{config}(s) = c$. We say that conditions φ and ψ are consistent (for ARD $(\mathcal{C}, \mathcal{T})$) if there exists a configuration that possibly satisfies $\varphi \wedge \psi$.*

We define $\mathcal{C}_{\mathcal{P},I,G}$ to be $\{c \in \mathcal{C} | \rho_{(\mathcal{P},Task(I,G))} < \infty\}$.

We are now ready to present the algorithm. The idea behind it is as follows. It takes as its input a task description and a joint KBP for n-ARU, and uses them to generate a progress measure over the configuration space of the ARD. It also (nondeterministically) assigns an ordering to the sub-ARUs. Next, it assigns to each sub-ARU its projection of the original joint protocol. It then sequentially goes through the sub-ARUs' protocols one at a time, adding to each just those additional knowledge conditions needed to ensure coordination with the preceding sub-ARUs.

Knowledge-Based Distribution Algorithm

Input: (!) A task *Task(I,G)* (2) A joint KBP *V* for n-ARU in disjoint form.

Output: distributed KBP for n-ARU.

1. Construct the progress measure $\rho_{(\mathcal{P},Task(I,G))}$.
2. Nondeterministically select an ordering over {1,...,n}.
3. Project *V* to each sub-ARU. Call the initial protocol of ARU*, V_k.
4. Let $r = 2$. Repeat the following until *r — N:*
   (a) Repeat sequentially for all knowledge conditions appearing in $\mathcal{P}_r$.
      i. Let ω be the current condition examined; let τ_r be the transition assigned to φ by $\mathcal{P}_r$.
      ii. Set *C — false*
      iii. Repeat for all (r - l)-tuples of transitions $\tau_1, \ldots, \tau_{r-1}$ that $ARU_1,...,ARU_{r-1}$ may perform under conditions consistent with *(alpha.*
         A. If there exist transitions $r_r+i,\ldots,r_n$ such that τ_i (where *r < i < n)* is assigned to- ARU, under a condition consistent with φ, and such that $(\tau_1, \ldots, \tau_r, \ldots, \tau_N)$ makes progress (according to $\rho_{(Task\ (I,G)))}$ on all configurations in $\mathcal{C}_{\mathcal{P},I,G}$ that possibly satisfy φ, then set *C* to
$$C \vee \bigwedge_{1 \le i < r} \theta_i$$
where θ_i is the knowledge condition in $\mathcal{P}_i$ under which τ_i is performed,
      iv. Replace the condition "φ *then* $c\tau$" in r's protocol by:
"$\varphi \wedge K_r C$ *then do* τ."
   (b) Collect conditions corresponding to the same transition (i.e., replace "$K_r\psi$ *then do* τ" and "$K_r\psi'$ *then do* r" by "$K_r(\psi \vee \psi')$ *then do* τ)."

Theorem 1 *Let V be a joint KBP with a finite number of executions from I and with conditions $\{K\varphi_i | i \in A\}$, where $a\varphi_i$ (i ∈ A)) are about C i$\mathcal{S}$[I]. V performs the input task Task(I, G) iff the distributed KBP generated by the algorithm performs Task(I, G).*

Several comments about the algorithm. First, different orders on the sub-ARUs will result in different distributed protocols. Roughly, the later a sub-ARU is in the ordering, the more it has to know, since it must coordinate its actions with the sub-ARU preceding it. Although we do not have a way of quantifying the 'ease' of each distribution, some ordering may result in requirements that are easier to achieve in practice than others. Second, the output of the algorithm is highly sensitive to the input task. A joint protocol usually embodies much more knowledge (i.e., coordination) than is actually needed for any given task. Our algorithm uses the progress measure, and hence the task, to determine just how much knowledge to add to each sub-ARU. Finally, an implicit assumption is that the change to a distributed protocol does not affect the function *Init.*

The distributed protocol we obtain is not necessarily minimal in its knowledge requirements and its properties, as well as those of other possible algorithms, deserve further investigation. However, it does offer a method for obtaining non-trivial specification of sufficient conditions for task distribution; these are the knowledge conditions of the distributed KBP obtained as its output.

Example 1 (cont.) The discussion in the previous section suggests the following KBP for our task: If *Kg* (Stop(armi),Move(arm2)); if $K\varphi$ (Move(armi),Stop(arm2)) where the proposition *g* and φ are defined as before.

Distribution. We start by running the algorithm on this protocol and the given task. We skip the construction of the progress measure, which is straightforward. After performing the first three steps we have:

Protocol for arm_1: K_1g t o p ; $K_1\varphi \wedge \neg K_1g$ ' e .

Protocol for arm_2: If K_2g Move; If $K_2\varphi \wedge \neg K_2g$ Stop.

Step 4 calls for revising arm_2s protocol. The first condition there is Kg, and both actions Stop and Move could be performed by arm_1 under this condition, since positions 3 and 4 satisfy both *g* and φ. The only action by arm_1 that makes progress when arm_2 does Move, is Stop. Stop is performed by arm_1 under the condition $K\backslash g$. Therefore, we must replace K_{2g} with $K_2g \wedge K_2K_1g$, which is equivalent to $K_2.K_{1g}$ Similarly, when we examine the action Stop, performed under the condition $K\varphi$, we find that it makes progress only when joined with Move. Therefore, we substitute the action $K_2(K_1\varphi \wedge \neg K_1g)$| f $K_2\varphi$.V e obtain:

Protocol for armi: If K_1g Stop; If $K_1\varphi \wedge \neg K_1g$ **Move**.

Protocol for arm_2: If K_2K_1g **Move**; If $K_2(K_1\varphi \wedge \neg K_1g)$ Stop.

Sensitivity of distribution to task. Consider the task of Example 1, but now the table is heat-resistant, so that the second arm need not start moving precisely when the first one has stopped, and in addition, the second arm may hit the object while still in motion. Clearly the above distributed KBP will do, since the original task is more stringent than this one, but the reader may verify that, as a result of the optimization performed by the algorithm, we obtain the following KBP:

Protocol for a If K_1g **Stop**; If $K_1\varphi \wedge \neg K_1g$ e .

Protocol If K_2g **Move**; If $K_2\varphi \wedge \neg K_2g$ Stop.

Sensitivity of distribution to the ordering of ARUs. We mentioned that the ordering of ARUs can make a difference. If arm_2 was given higher priority than arm_1, we would have

obtained the symmetric protocol:

Protocol for arm_1: If $K_1 K_2 g$ Move; If $K_1(K_2\varphi \wedge \neg K_2 g)$ Stop.

Protocol for arm_2: If $K_2 g$ Stop; If $K_2\varphi \wedge \neg K_2 g$ Move.

In a precise sense, which we do not have space to explain, both distributions embody the same amount of knowledge. However, from the pragmatic point of view one protocol might be vastly superior to another. For example, in some situations it may be easier to detect goal configurations from a distance. In that case the second protocol may be more appropriate, since detecting *g* would be easier for arm_2.

4 Discussion

This paper makes two main contributions:

- We present a formal model of robotics domains, a logical language for reasoning about the model, and define natural measures of the information complexity of robotics tasks and the informational abilities of robots based on the notion of knowledge.
- We use the model and the language to provide a distribution algorithm for knowledge-based robotic protocols.

Information, and more specifically sensing, are issues of central importance in robotics. The unreliability of sensors and the difficulty in reasoning about sensory information has lead some researchers to attempt to reduce the sensing requirements of tasks (e.g.,[Erdmann and Mason, 1993; Goldberg, 1993]. More generally, there has been increasing interest in understanding the inherent sensing requirements of a task. [Erdmann, 1994] is one such effort which has had major influence on our work. Erdmann argues that sensors should be constructed to provide sufficient information to choose an action that makes progress. He uses progress measures, discussed earlier, to do so. Erdmann also talks about knowledge, and the semantics of his notion of information is very similar to the standard semantics of knowledge which we use. One contribution of our work is to supply a formal language that captures these ideas, which (as we have shown) has natural extensions.

Another influential idea is Donald's notion of information invariants [Donald, 1994], in which 'reductions' play an important part. Roughly speaking, system A can be reduced, to system B if we can use system B to obtain the same information that system A can obtain. In a longer version of this paper we discuss a natural notion of knowledge-based reduction that arises in our work, and tie it in with Donald's work.[5]

Motivating our investigation of task distribution was the work of [Donald *et a*/., 1993]. This work examines distribution of manipulation tasks, starting from a centralized protocol. Their analysis of the information needed there shows that it pertains to the configuration of the manipulating robot w.r.t. the manipulated object. Using our terminology, we can say that from their work it can be concluded that these tasks do not require two levels of knowledge for their distribution (i.e., one robot does not need to know the state of the other). They are then able to obtain an asynchronous distributed protocol for such problems.

Finally, the use of prioritization in multi-robot environments appears in a different context in [Erdmann and Lozano-Perez, 1987]. There, prioritization is used to produce motion plans for multiple moving objects.

We believe that the marriage of knowledge representation and robotics is an extremely promising direction, which we intend to pursue.

Acknowledgment: We are grateful to Bruce Donald for numerous insightful discussions and comments, to Joe Halpern for many important suggestions and corrections, and to Nir Friedman for comments on previous drafts. This work was partially supported by AFOSR and NSF grants AF F49620-92-J-0547 and IRI-9220645.

[5] Updated versions of this work appear in URL http://robotics.stanford.edu/users/brafman/bio.html

References

[Brafman *et al.,* 1994] R. I. Brafman, J. C. Latombe, Y. Moses, and Y. Shoham. Knowledge as a tool in motion planning under uncertainty. In R. Fagin, editor, *Proc. 5th Conf. on Theor. Asp. of Reas. about Know.,* pages 208-224. Morgan Kaufmann, 1994.

[Canny, 1989] J. F. Canny. On computability of fine motion plans. In *Proc. of the 19S9 IEEE Int. Conf. on Rob. and Aut,* pages 177-182, 1989.

[Donald *et al,* 1993] B. R. Donald, J. Jennings, and D. Rus. Information invariants for cooperating autonomous mobile robots. In *Proc. of Int. Symp. on Robotics Research,* 1993.

[Donald, 1994] B.R. Donald. On information invariants in robotics. *Artif Inteli,* 72(1):217- 304, 1994.

[Erdmann and Lozano-Perez, 1987] M. Erdmann and T. Lozano-Perez. On multiple moving robots. *Algorithmica* 2(4):477-521, 1987.

[Erdmann and Mason, 1993] M. Erdmann and M.T. Mason. An exploration of sensorless manipulation. In *IEEE Int. Conf. on Rob. and Aut.,* 1993.

[Erdmann, 1994] M. Erdmann. Understanding actions and sensing by designing action-based sensors. In *IEEE ICRA Workshop on Design,* 1994.

[Fagin *et ai,* 1995] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about Knowledge*. MIT Press, 1995.

[Goldberg, 1993] K.Y. Goldberg. Orienting parts without sensors. *Algonthmica,* 10(2):201-225, 1993.

[Halpern and Moses, 1990] J. Y. Halpern and Y. Moses. Knowledge and common knowledge in a distributed environment. *J. ACM,* 37(3):549-587, 1990.

[Lozano-Perez, 1983] T. Lozano-Perez. Spatial planning: a configuration space approach. *IEEE Trans, on Comp.,* C-32(2):108-120, 1983.

[Moore, 1985] R. C. Moore. A formal theory of knowledge and action. In *Formal Theories of the Common Sense World,* 1985.

[Rosenschein, 1985] S. J. Rosenschein. Formal theories of knowledge in AI and robotics. *New Generation Comp.,* 3:345-357, 1985.