Available online at www.sciencedirect.com
SCIENCE DIRECT®
Robotics and Autonomous Systems 49 (2004) 13–23
Robotics and Autonomous Systems
www.elsevier.com/locate/robot

Integrating cognition, perception and action through mental simulation in robots

Nicholas L. Cassimatis[a,*], J. Gregory Trafton[b,1], Magdalena D. Bugajska[b,2], Alan C. Schultz[b,2]

[a] *Department of Cognitive Science, Rensselaer Polytechnic Institute, Troy, NY 12180, USA*
[b] *Naval Research Laboratory, Code 5515, 4555 Overlook Avenue SW, Washington, DC 20375-5337, USA*

Received 27 July 2004; accepted 27 July 2004

Abstract

We argue that many problems in robotics arise from the difficulty of integrating multiple knowledge representation and inference techniques. We describe an architecture that integrates disparate reasoning, planning, sensation and mobility algorithms by composing them from strategies for managing mental simulations. Since simulations are conducted by modules that include high-level knowledge representation and inference techniques in addition to algorithms for sensation and reactive mobility, cognition, perception and action are continually integrated. An implemented robot using this framework in object-tacking and human–robot interaction tasks demonstrates that knowledge representation and inference techniques enable more complex and flexible robot behavior.

Keywords: Cognitive robotics; Knowledge representation; Cognitive models

1. Introduction

We propose that many problems in designing intelligent autonomous robots arise from the difficulty of combining knowledge representation and inference techniques into one robot that can construct and maintain rich, coherent and dynamic models of its environment. These include the problems involved in data fusion, symbol grounding and flexibly combining reasoning, planning, perception and action.

* Corresponding author. Tel.: +1 518 276 3853; fax: +1 518 276 3017.
E-mail addresses: cassin@rpi.edu (N.L. Cassimatis), trafton@itd.nrl.navy.mil (J.G. Trafton), magda@aic.nrl.navy.mil (M.D. Bugajska), schultz@aic.nrl.navy.mil (A.C. Schultz).
[1] Fax: +1 202 404 4080.
[2] Fax: +1 202 767 3172.

0921-8890/$ – see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.robot.2004.07.014

The data fusion problem involves integrating information from multiple sensors into a coherent model of the environment. For example, when sensor A in a robot detects an object and sensor B in the same robot detects an object, the robot must determine whether the two sensors are detecting the same or different objects. Since each class of sensor information has its own best representation, sensor fusion is in part a problem of combining multiple representation schemes. Since the decision about whether information from two sensors is about the same object can depend on world knowledge (e.g., that objects of a particular category are too slow to have moved from sensor A's range to sensor B's range in so short time) that is often best represented using artificial intelligence reasoning techniques, robust sensor fusion also requires the integration of sensor information with traditional AI representations.

Some researchers (e.g. [1]) have argued that symbols manipulated by traditional artificial intelligence algorithms cannot have any relation to objects, events and relations in the environment that are perceived through sensors yielding data in very different representations. This "symbol grounding" problem is fundamentally a problem of integrating different sorts of ("perceptual" and "symbolic") representations.

A third problem in robotics stems from the need to use multiple knowledge representation techniques to model the world. It is well known that different aspects of the world are best represented using different knowledge representation techniques. Since different representations often require or are associated with different inference algorithms (e.g., rule matchers operate on rules, junction tree algorithms operate on Bayes Networks) robots must integrate multiple reasoning and planning algorithms with each other. This is a difficult problem in robotics and artificial intelligence generally since AI algorithms can have such different data and control structures. It is not at all clear, for example, how to tightly integrate a STRIPS [2] planner with a localization algorithm based on POMDPs [3].

Behavior-based and reactive (e.g. [4,5]) robotics researchers have reacted to this difficulty by dispensing with algorithms that are based on constructing and manipulating internal representations that are not closely (in time and space) tied to what is being immediately sensed. Since there is no "old" information and the robot never needs to pause to perform computations on internal representations, these systems appear to have the virtue of always taking the most relevant action given the current information. A major problem with these approaches is that so much of what determines the correct action for a robot to take involves the environment in past or future, occluded, spatially distant and/or hypothetical situations that cannot be immediately sensed. For example, the future consequences of an action (i.e., the future state of the hypothetical world in which that action is taken) help to determine whether a robot should take that action. For tasks or environments with all but the most minimal complexity, one cannot anticipate all possible classes of sensor readings and precompile an appropriate reaction for each situation. Thus, representation-free approaches are severely limited by the complexity of the environments they can deal with and the tasks they can achieve.

Since so many problems in robotics involve the integration of multiple representation and inference techniques, we have developed a robotic architecture that supports the combination of these techniques. The architecture, which we call Polybot, is based on the Polyscheme cognitive architecture [6] for solving integration problems in artificial intelligence and cognitive science generally. Polyscheme differs from traditional cognitive architectures based on one or a few data structures by enabling inference based on multiple data structures. Polyscheme differs from the many multi-agent system architectures that encapsulate specialized algorithms in modules by enabling every step of every algorithm to be executed using multiple representations and be potentially assisted by every other algorithm.

The next two sections explain how Polybot composes reasoning and planning techniques from cognitive science and artificial intelligence research by sequences of mental simulations. Since Polybot enables these sequences of simulations to be interleaved and since a particular simulation can be part of the execution of more than one algorithm, multiple reasoning and planning algorithms are easily integrated. Since each simulation is conducted using multiple representations, spanning the continuum from low-level perception and mobility techniques to higher-level knowledge representation schemes, reasoning and planning in Polybot are continuously integrated with perception, action and multiple knowledge representation schemes. The final section gives a brief overview of

the initial results we have achieved with this approach in object tracking and human–robot interaction tasks.

2. An architecture for integrating multiple representation and inference techniques through the control of mental simulations

Polyscheme is motivated by the view that the difficulty of achieving the benefits of both high-level reasoning and planning and behavior-based and reactive systems is a problem of combining different representation and inference techniques into one system. In using Polyscheme to design Polybot, we aimed to create an architecture that could engage in reasoning and planning that is at every step responsive and adaptive to information from (potentially noisy) sensors and changes in the world.

Polyscheme's fundamental approach to this problem is to implement and execute reasoning and planning algorithms using mental simulations that are based on perceptual and reactive representations as well as traditional artificial intelligence representations. Polyscheme takes "reactive" components that typically choose their actions only with reference to the currently sensed state of the world, and allows them to "react to" represented or simulated states of the world. Using strategies for choosing which simulations to run (i.e., what time, place or hypothetical world to simulate), Polyscheme implements high-level AI algorithms by executing mental simulations. Because simulations are composed in part of reactive and perceptual subcomponents, reasoning is constantly and thoroughly integrated with perception and action. We now describe how these ideas are realized in Polybot.

2.1. Polyscheme encapsulates mobility and perception techniques in specialists

Polyscheme encapsulates the functionality of robot perception and mobility techniques in modules called *specialists*. Because our experience has shown that multiple techniques for mobility and perception are useful, specialists may use any algorithm or data structure to implement their functionality. For example, Polybot includes a specialist that identifies the location and category of objects using color segmentation [7]. All the inferences and actions in Polyscheme are executed by specialists. The rest of the Polyscheme architecture is aimed at coordinating and sharing information among specialists.

2.2. Specialists communicate using a representation-neutral language based on a common ontology

Specialists must share information in order to perform their functions. For example, a mobility specialist will need information from a perception specialist in order to avoid obstacles. In order for specialists to be able to communicate such information, all Polyscheme specialists use the same common language to communicate information. Because the language is only used for communication and not computation, we focused on a simple, though expressive, propositional language. Specialists in Polyscheme must be able to translate between their internal data structures to this propositional language. In order for these specialists to communicate, the language must adhere to a standard known to all specialists. For example, an object recognition specialist indicates that a cone is at location `p` at time, `now`, with the propositions `Location(o,p,now)`, `xOf(p,3.4,now)`, `yOf(p,4.2,now)`, and `Category(o,Cone,now)`. If other specialists are to use this information (e.g., a mobility specialist that knows to avoid cones), then they must share predicates such as `Location`, `xOf`, `yOf` and `Category` and understand the units of distance and time used in their arguments. Specialists are not committed to using these units and these predicates in their own internal computations or even using representations based on predicates on objects. They must simply be able to *translate* between this representation and their own *internal* representation. By separating the representation used for communication from the representation used for inference, we achieve the benefits of a common ontology without the rigidity often associated with such knowledge representation schemes.

2.3. Specialists implement a common set of functions used to share information

Specialists each implement a common set of functions that Polyscheme uses to coordinate information flow and behavior. By standardizing this set of functions and by using the common propositional language

described in the last section, specialists do not need to take into account the implementation details of other specialists and Polyscheme can be extended with new specialists more easily. These functions are as follows:

`ReportOpinion(prop,otherSpecialist, tv)`. A specialist learns that `otherSpecialist` believes that the proposition, `prop`, has truth value `tv`.

`StanceOn(prop)`. Returns the truth value the specialist believes `prop`. These truth values are annotated with the degree of confidence the specialist has in that the `prop` being true or false.

`RequestedFoci( )`. Returns a set of propositions that the specialist would like to focus on. These include, but are not limited to, propositions that the specialists wants to assert as being true or false and subgoals (i.e. propositions whose truth values would help the specialist take a more accurate stance on another proposition).

`Groundings(prop)`. Returns a set of closed propositions that ground the open (i.e., its arguments have open variables) proposition, `prop`.

2.4. Specialists must alert the system to changes in their beliefs

Because robotic sensors are noisy and because they generally yield incomplete information about the environment (because of limited sensor ranges and occluders), Polyscheme's specialists will at least occasionally change their stance on a proposition. When this occurs, it is important that other specialists are notified of this so that they do not continue to make inferences and take actions based on bad information. A specialist's `RequestedFoci( )` function must therefore include among its assertions stances on propositions on which it has revised its beliefs.

2.5. Specialists must be able simulate non-immediate states

Since robots' actions and inferences often depend on past, future, distant and/or hypothetical situations, specialists in Polyscheme must be able to react not just to the immediately sensed state of the world, but some representation (described in the next section) of past, future, distant, invisible or hypothetical states. We call any state, event, region, or situation that is not currently sensed (because it is distant, occluded, hypothetical at another time and/or at another place) *non-immediate*. To represent non-immediate situations, Polyscheme's ontology includes temporal intervals and hypothetical worlds. The last two arguments of every proposition input to and output from specialists are, respectively, a time and world argument. For example, the proposition that object, `o`, is located at point `p`, in hypothetical world, `w`, at time `t` is indicated: `Location(o,p,t,w)`. In the next section we discuss how Polyscheme represents non-immediate states.

2.6. Specialists simulate non-immediate states using multiple representations

If specialists must react to non-immediate states of the world, then there must be some representation of those states that is different from the current state of the robot's sensors. There must be a memory for past events, properties of objects and relations between them; there must be a way to represent future and hypothetical states so their desirability can be evaluated and it must be possible to represent distant or occluded parts of the environment. Because different representational techniques are most appropriate for different aspects of the world (for example, temporal constraint graphs for temporal intervals, spatial maps for object locations) Polybot includes multiple specialists that encapsulate representations for different aspects of the world. Representations that have already been implemented include spatial maps, temporal constraint graphs, scripts and directed graphs.

Given that robots must represent non-immediate states, how do they decide what the truth is about those states? There are several mechanisms for accomplishing this. Memory is a mechanism for deciding what was true in the past. Causal rules, constraint satisfaction algorithms, and dynamic simulation can predict what is true in future and/or hypothetical states. They can also decide what is happening in occluded regions of the environment, for example, when the causal rule specialist predicts that in the absence of an obstacle or external force, a ball rolling behind an occluding object will continue to exist and roll behind the occluder even though the ball and its motion are not detected at that moment by the robot's sensors. We use Minsky's [8] term *simulus* to refer to the input to specialists from a simulated world.

Although inferential specialists use high-level AI algorithms, the way they interface with the rest of the system has a reactive character because they implement the same specialist functions as reactive specialists. For example, Polyscheme's category hierarchy specialist learns new information about an object through its `ReportOpinions( )` function, "reacts" to it by using its own internal data structures and algorithms to make inferences about the category membership of the object, and uses the `RequestedFoci( )` function inform other specialists of these.

Table 1 illustrates how several "high-level" artificial intelligence algorithms take on a reactive character when encapsulated in specialists.

2.7. Specialists must focus their attention

All specialists in Polyscheme react to the same proposition at the same time. There are two reasons for this. First, a surprisingly large number of specialists are relevant to any particular proposition. For example, when inferring whether a falling object will continue to fall, a causal specialist will predict that it will do so if the region underneath the object is empty, the perceptual specialist might be able to see whether there is a supporting object, the object location specialist might be able to remember if there was a support, etc. Second, if a specialist acts on an inferred or perceived proposition before checking other specialists' stances on it, the specialist might be acting on incorrect information that can lead to harmful mistakes or at best require a time-consuming retraction of incorrect actions and inferences.

For these reasons, at every time step in Polyscheme, specialists focus on the same proposition. Once a proposition, `P`, is chosen (as described in the next section), the following sequence occurs:

- Polyscheme calls `StanceOn(P)` for each specialist to determine the consensus truth value of all the specialists on `P`.
- For each specialist, Polyscheme calls the function `ReportOpinion(P,specialist,tv)` to report specialists' truth values for `P` to each other.
- Polyscheme calls `RequestedFoci( )` to get propositions the specialists would like to focus on soon.

The chief form of communication among specialists in Polyscheme is through the focus of attention. Through functions such as `StanceOn( )`, `ReportOpinions( )`, `Groundings( )` and `RequestedFoci( )`, specialists communicate information to and request information from each other. When a proposition becomes the focus of attention, each specialist learns the other specialists' opinions of its truth and has an opportunity to ask questions that flow from or assert propositions that follow from the focal proposition.

2.8. Specialists must quickly react to the focus

Because the environment can change quickly or because new sensor information may become available at any moment, specialists must quickly execute functions such as `StanceOn( )` and `ReportOpinion( )` so that the specialist can constantly focus on and make inferences with the newest information. This is an important component of Polyscheme's solution to the problem of inferences and plans that are invalidated before they are completed or executed.

2.9. Simulations result from the focused attention of specialists

One result of the architectural principles discussed in this section is that Polyscheme's specialists collectively perform a kind of simulation. When Polyscheme focuses on a particular time and world, the architecture forces all specialists to focus on that time and world. The sum effect of this attention will be that specialists will make inferences about that world and therefore elaborate Polyscheme's representation of it. Since some of these inferences will involve the consequences of states and events in this world, Polyscheme will focus on subsequent times in the same world. The result is that specialists will perform a dynamic simulation of that world.

2.10. Focus schemes guide simulation

Nonrepresentational robotic systems must constantly (implicitly or explicitly) make a choice, "Where do I look now?", because sensors are inherently directed towards a local region of space. Even sensors which uniformly monitor the region surrounding a

Table 1
Reactions to simuli in multiple representations

Simulus	Representation/algorithm	Action
Sensors	None for reactive systems	Action
Cause	Causal rules/rule matching	Assert effect
Input layers	Neural networks/network propagation	Output layer
Object category information	Category hierarchies/graph walking	Assert other categories the object belongs to
Two temporal intervals	Temporal intervals/constraint propagation	Assert the possible constraints between the two intervals

robot can be directed to another region by the robot's motion. For Polybot, the number of choices is even greater because specialists can focus on many possible simulated worlds in addition to the immediate world itself.

As indicated in the previous section, when Polyscheme specialists focus on a proposition, they ask for a set of propositions to focus on through their `RequestedFoci( )` function. Polyscheme's *focus manager* chooses from these requests (based on their level of urgency (as indicated by the specialist) and several other factors). If the proposition is open or "ungrounded", i.e., if it contains an open variable, the focus manager chooses a proposition that grounds the proposition. Once the focal proposition is chosen, Polyscheme calls the various specialist functions as indicated in the last section.

Polyscheme thus continuously chooses a proposition to focus on, allows specialists to communicate about and make inferences about this proposition and then chooses the next proposition to focus on based on specialists' requests. Through their ability to request Polybot to focus on a proposition, specialists can influence the flow of attention and hence computation. We call different strategies for guiding attention *focus schemes*. We have already encountered one focus scheme implemented by all specialists:

Resimulation focus scheme. When a specialist infers that a proposition `P` has a truth value that is the opposite of the truth value it returned during the last call of `StanceOn(P)`, include `P` in the return set of `RequestedFoci( )` the next time that function is called. Less formally, when a specialist changes its stance on `P`, it should request that `P` be focused on again.

Another example is the prediction focus scheme. It tells Polybot to simulate the results of an action before executing it:

Prediction focus scheme. When the motor specialist is about to take an action, `A`, simulate `Occurs(A,t,w)` where `t` is the next time step and `w` is the hypothetical world in which `A` is taken at time `t`.

When the system focuses on `Occurs(A,t,w)`, all of the specialists in the system will infer what else is true in that world, i.e., what the consequences are of the action `A`, and request that these consequences be focused on through their `RequestedFoci( )` function. If `w` is a world that contains damage or harm, the motion specialist will not execute `A`.

3. Algorithms are implemented by strategies for choosing simulations

The most fundamental point of this paper is that many "high-level" artificial intelligence algorithms can be implemented by focus schemes for choosing simulations that Polyscheme's specialists execute. We illustrate this by showing how to implement backtracking search with the counterfactual simulation focus scheme.

Counterfactual simulation. When uncertain about `A`'s truth value (because of a lack of information or because of conflicting information), simulate the world in which `A` is true and the world in which `A` is false.

If when simulating one of these worlds, say where `A` is true, one of the specialists infers a fact that contradicts what is already known for certain, then the world where `A` is true is contradictory and hence `A` can be inferred to be false in the real world.

Consider the case where Polybot is uncertain of two propositions, `A` and `B`. In the simulated world in which `A` is true, there is still uncertainty about `B`. Thus the counterfactual simulation focus scheme still applies and imagines the world in which `A` and `B` are true and the world in which `A` and not-`B` are true. When one of

Table 2
Inference algorithms and the focus schemes simulations that implement them

Algorithm	Focus scheme that implements it
Case-based reasoning	Memory-based simulations
Prediction	Forward simulation
Counterfactual reasoning	Counterfactual simulation
Backtracking search	(nested) Counterfactual simulation
Backward chaining/ theorem proving	Antecedent simulation
Truth maintenance	Resimulation
Bayesian inference	Stochastic simulation

these leads to a contradiction, that world is not simulated any longer. Thus, the counterfactual simulating focus scheme can lead to nested simulations which effectively implement backtracking search.

Table 2 lists several focus schemes that implement important artificial intelligence algorithms. The resimulation focus scheme of the last section implements a form of truth maintenance since the result of focusing on a proposition whose truth value has changed will be to resimulate events and states the proposition relates to and hence change incorrect inferences based on the initial false belief. Cassimatis [9] has shown that the *antecedent simulation* (roughly, "when P implies Q and you want to know Q, simulate the world where P is true") implements a form of resolution theorem proving. The *stochastic simulation* focus scheme ("when P has probability $m/(m+n)$, simulate the world where P is true m times and the world where P is false n times) implements an approximate form of Bayesian inference that has been used widely by the uncertain reasoning community. Finally, *memory-based simulation* ("when G is a goal, simulate in the current situation actions you have previously taken that have achieved goals similar to G") implements a form of case-based reasoning.

4. Combining mental simulations resolves many integration issues in robotics

Now that we have described Polyscheme's approach to supporting multiple representation and inference schemes, we discuss how this helps to resolve many problems in robotics. We do so by presenting an example, illustrated in Fig. 1, which will illustrate several of this paper's themes.

4.1. An extended example

In 1a, Polybot starts by observing an orange robot. Its task is to track that robot. In 1b, Polybot sees that robot move behind a screen and loses visual contact with it. In 1c, Polybot sees an orange robot move out from the screen and infers that it is the robot it is tracking. As it moves towards that robot, it sees a barrier behind the screen (1D) that the robot could not have rolled through and infers that the orange robot it sees now is different from the robot it is tracking and goes to the other side of the obstacles to find that robot. Table 3 traces Polybot's focus during this scenario.

The following sections use this example to illustrate how using Polyscheme to combine multiple representation and inference techniques helps to resolve many of the problems surrounding building flexible robots that can engage in high-level reasoning.

4.2. Simulations are a medium for integrating multiple representations

The example shows that sharing information between different representations is fairly straightforward in Polyscheme. Each specialist has its own representation and can translate back and forth between it and the representation-neutral language used to encode the focus. For example, during step 9, the perception specialist perceives that `p2` is empty and encodes that in its own perceptual representation, returns `true` as a stance and the other specialists learn of this through their `ReportOpinion()` function. One of these specialists is the causal specialist which encodes that `p2` is empty in its causal rule language. Thus, through the focus of attention and the representation-neutral language, specialists can share information easily.

In this context, symbol grounding is less puzzling. The causal rule specialist can manipulate symbols representing emptiness without fear of losing touch with "physical reality" because each simulation it performs combines (through the focus of attention) information from sensors and information from the rule specialist. This is made possible because of the representation-neutral language the specialists share. Thus, our solution to the symbol grounding problem is to integrate

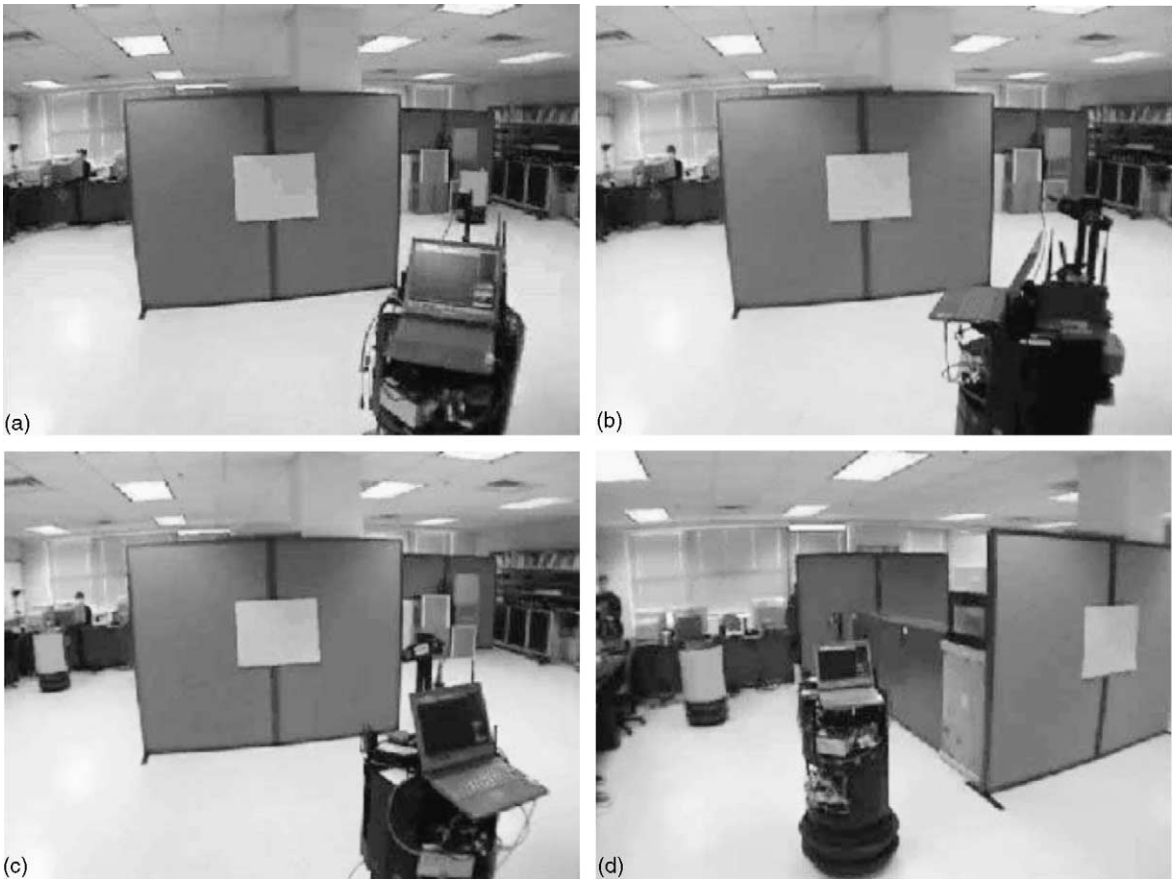

Fig. 1. Polybot tracks another robot as it moves behind an obstacle. The tracked robot moves from the right of the obstacle (a), behind the screen (b) and a robot that appears to be the tracked robot moves out from the left of the obstacle (c). Moving towards this robot (d), Polybot receives new sensor information which triggers a combination of belief revision and physical reasoning to conclude that this is not the originally tracked object.

perceptual specialists into the focus of attention so that each simulation and hence each step of every inference algorithm is constantly integrated and synchronized with perception.

4.3. Combining simulations combines algorithms

In this example, Polybot executes three different algorithms: prediction, backtracking (path) search and truth maintenance. Each is composed of several foci. Prediction is composed of foci 4–7 and 11–13, back tracking search is composed of 6, 9 and 10 and truth maintenance is composed of 9–13. Note that each algorithm's foci overlap with the others. This overlap or sharing of foci is a key to combination of algorithms in Polyscheme. When, for example, Polybot focuses on `Empty(p2,E,R)` in step 9, the specialists' inferences about whether that proposition is true are shared by both the search- and truth-maintenance algorithms. In general, since algorithms implemented by focus schemes are merely composed of foci, the data they operate on resides in the specialists which make inferences on the foci. Since foci can be shared by any algorithm, sharing information between these algorithms is simple.

4.4. Simulations integrate reasoning, planning, perception and action

Inference algorithms are composed of sequences of simulations executed by specialists. Since these spe-

Table 3
A trace of Polybot's focus

	Focus	Simulation	Algorithm	Explanation
1	–	Immediate		Perception specialist observes the cart move from behind the boxes (Location(robot1,p1,t1,R) and Category(robot1,Cart,t1,R)) and then observes a cart come out from behind the boxes (Location(robot2,p3,t3,R) and Category(robot2,Robot,t3,R))
2	Category(robot2,Robot,T1,R)	Immediate		Identity hypothesis specialist's neural network infers that Same(robot1,robot2,E,R) and requests focus for that proposition
3	Same(robot1,robot2,E,R)	Immediate		The difference specialist detects a difference in the location of tracked cart, infers an change event and requests for a focus on Exists(d,E,R), Category(d,ChangeEvent,E,R), etc.
4	Category(d,Change,E,R)	Recent	Prediction	The causal rule specialist infers that since p1 and p3 are not adjacent points, there must be an intermediate point, P, that robot1 visited and that P must be empty
5	Occupant(P,nothing,t2,R)	Recent	Prediction	The space specialist infers that there P might be the intermediate point and requests for a focus on Same(p2,P,w), where w is the world where p2 and P are the same
6	Same(p2,P,E,R)	Immediate hypothetical	Prediction, Search	The tracking specialist infers that since the location robot1 is now at P that the system should move there
7	–	Future hypothetical	Prediction	The prediction focus scheme simulates that motion and finds no problems. (This takes several steps.)
8	–	Immediate		While moving towards that location, the perception specialist sees p2 for the first time and sees that P is not empty, but Occupant by a wall
9	Occupant(P,wall,t2,R)	Current hypothetical	Search, Truth maintenance	The difference specialist notices the difference between Occupant(P,wall,t2,R) and not Occupant(P,nothing,t2,R). Since p2 is certainly empty, and P is certainly not empty, then the difference specialist assumes that in fact Same(p,p2,E,R) is false and requests focus for that proposition because of the resimulation focus scheme
10	Same(p2,P,E,w)	Hypothetical	Truth maintenance, Search	The space specialist cannot find any other places that P might be equal to and thus assumes it does not exist and request focus for Exist(P,E,R)
11	Exists(P,E,R)	Immediate	Truth maintenance, Prediction	Since no intermediate point exists, the causal rules specialist retracts the existence of the event that implies it and request focus on it because of resimulation
12	Exists(d,E,R)	Past	Truth maintenance, Prediction	Since delta is retracted, the identity of Same(robot1,robot2,t2,E,w) is retracted
13	Same(robot1,robot2,E,R)	Immediate	Truth maintenance, Prediction	Thus, robot1 and robot2 are different and robot1 is still behind the box on the left
14	–	Immediate		Motion specialist initiates movement in that direction

cialists include specialists for perception and mobility, the combination of high- and low-level computational techniques is constant in Polybot. For instance, the focus in step 9 of the example is part of a path search. The focus, `Empty(p2,E,R)`, is perceived to be false and modifies the course of the path search. This is a simple example of a system's sensors being able to influence the course of a high-level artificial intelligence algorithm as it is being executed.

4.5. Reasoning and planning with information from noisy sensors in a dynamic world

Three features of this approach greatly reduce the tension between the flexibility of representation-free, reactive systems and the power of high-level artificial intelligence algorithms. First, since algorithms are composed by simulations executed by specialists and because these specialists include perceptual specialists, every step of inference is always being checked against new sensor information. Thus, revisions in sensor readings or changes in the world will be detected immediately. Second, because each specialist is obligated to immediately broadcast these changes to the rest of the specialists as soon as they occur, inference can be adjusted immediately. Finally, because algorithms are composed of reactions that are required to follow quickly from simuli, there is no long lag between the formulation and execution of a plan during which the plan can become invalidated by changes in the world.

In the example above, as soon as Polybot's initial assumption that `P` is empty is seen to be false in step 9, that change is broadcast to the rest of the system and search is immediately altered. The revision does not need to wait until the end of prediction and truth maintenance in step 13 for it to influence inference.

4.6. Results and conclusion

We have developed this framework and demonstrated its benefits primarily in object tracking and human–robot interaction domains. This work has been based on several specialists we implemented, including those for perception (which uses CMVison), movement (a reactive planner), temporal constraints ([10], temporal intervals), spatial location (cognitive maps), state change detection, causation (production rules), ontology (category lattice), uncertainty, perspective and object identity (neural networks).

Many tasks require robots to keep track of the identity and location of at least some objects in their environment. In general, the sensors of a robot tracking an object detect multiple objects each instant. The robot must decide which of these sensor readings belong to the object it is tracking. When the distance in time and space from one object sighting to the next is brief, this problem is often relatively easy and there are many algorithms for solving it. However, when objects are occluded from the robot for long periods of time, the problem is much more difficult because of the potentially large number of interactions the occluded object could have participated in. For example, the last section described an object tracking scenario where Polybot was able to rule out a match between two robot sightings by searching for a continuous path between the two sightings and reasoning about the physical interactions those involved. Thus far, this had been difficult for many object tracking systems because they had only been based on algorithms not well-suited for physical reasoning and path search whereas Polybot can use Polyscheme to combine physical reasoning and path search with the short-distance object-tracking algorithms in its vision specialist. By combing both classes of techniques, Polybot's object tracking has been greatly improved. Polybot's visual system itself could only track an object when it is occluded for less than a meter and less than a second. Using Polyscheme to combine this ability with physical reasoning, Polybot can now often track objects even when they are occluded for several meters and for longer than 1 min.

In our human–robot interaction work, Polybot has been able to improve upon previous work on an object reference tasks by using its ability to simulate the world from a person's perspective in order to more accurately understand and predict his actions. We created a task in which a human would use language to refer to an object and it was the robot's task to go to that object. Since there were many instances of the same object in the room and since humans could see objects that the robots could not, and vice versa, many of the human's utterances were ambiguous when taken literally. By using Polybot's simulation abilities, robots were able to take the perspective of the human in the task and significantly improve the accuracy of their understanding.

One limitation of our approach so far has been the difficulty of formally characterizing the conditions under which a robot designed using this approach is correct or optimal. This characterization is difficult to achieve because Polyscheme combines techniques based on different formal frameworks, which do not as of yet have a single framework that subsumes them, and can include techniques for which there is no formal framework. In return for this sacrifice, Polyscheme enables systems that include functionality not available from any particular computational technique and enables these techniques to be flexibly combined with perception and action.

We believe that these results are a first step towards demonstrating that this approach towards integrating kind of knowledge representations and inference algorithms reduces the tension between achieving complex reasoning and planning using high-order knowledge representation and inference techniques and maintaining the flexibility of reactive architectures.

Acknowledgement

Derek Brock helped to design the tasks we used in our human–robot interaction work. Jamie Lennon helped integrate Polyscheme and Polybot's vision and motion planning subsystems.

References

[1] S. Harnad, The symbol grounding problem, Physica D 42 (1990) 335–346.
[2] N. Fikes, N. Nilson, STRIPS: a new approach to the application of theorem proving to problem solving, Artificial Intelligence 2 (3–4) (1971) 189–208.
[3] L.P. Kaelbling, M.L. Littman, A.R. Cassandra, Planning and acting in partially observable stochastic domains, Artificial Intelligence (1998).
[4] R.A. Brooks, Intelligence without representation, Artificial Intelligence 47 (1991) 139–159.
[5] P. Agre, D. Chapman, What are plans for? Journal for Robotics and Autonomous Systems 6 (1990) 17–34.
[6] N.L. Cassimatis, Polyscheme: a cognitive architecture for integrating multiple representation and inference schemes, Ph.D. dissertation, Media Laboratory, Massachusetts Institute of Technology, 2002.
[7] V. Rehrmann, L. Priese, Fast and robust segmentation of natural color scenes, in: R., Chin, T.-C., Pong, (Eds.), Computer Vision—ACCV98, vol. I, Hong Kong, China, 1998, pp. 598–606.
[8] M.L. Minsky, The Society of Mind, Simon and Schuster, New York, 1986.
[9] N.L. Cassimatis, A framework for answering queries using multiple representation and inference techniques, in: Proceedings of the 10th International Workshop on Knowledge Representation Meets Databases, 2003.
[10] J.F. Allen, Maintaining knowledge about temporal intervals, Communications of the ACM 26 (11) (1983) 832–843.

Nick Cassimatis is an Assistant Professor of Cognitive Science at the Rensselaer Polytechnic Institute. In his research, he designs intelligent systems and cognitive models that combine the benefits of multiple kinds of computational and representational mechanisms. He received his PhD from MIT's Media Laboratory in 2002 and was a National Research Council postdoctoral associate at the U.S. Naval Research Laboratory through the summer of 2004.

J. Gregory Trafton is a researcher at the Navy Center for Applied Research in Artificial Intelligence. In addition to cognitive robotics, his research interests include the cognition of complex visualizations, interruptions, graph comprehension, cognitive modeling, problem solving, tutoring systems, learning, transfer and the interaction between learning environments and human-computer interaction. He received his PhD from Princeton University.

Magda Bugajska is a computer scientist at the Navy Center for Applied Research in Artificial Intelligence. She received a BS in Mathematics and Computer Science with minor in AI and Robotics from Colorado School of Mines and is currently finishing up a MS degree program in Computer Science at George Mason University. Her research interests include evolutionary computation, cognitive modeling, and robotics.

Alan C. Schultz is the Head of the Intelligent Systems Section, Navy Center for Applied Research in Artificial Intelligence at the Navy Research Laboratory in Washington, DC. His research is in the area of evolutionary computation, evolutionary robotics, learning in robotic systems, robot/human interfaces, and adaptive systems. He is the recipient of an Alan Berman Research Publication Award, and has published over 40 articles on machine learning and robotics. Alan is currently the co-chair of the AAAI Symposium Series, and chaired the 1999 and 2000 AAAI Mobil Robot Competition and Exhibitions.