Contention Scheduling: A Viable Action-Selection Mechanism for Robotics?

Virgil Andronache **Matthias Scheutz**
Department of Computer Science and Engineering
University of Notre Dame
Notre Dame, IN 46556
{vandrona|mscheutz}@nd.edu

Abstract

Contention scheduling (Cooper & Shallice 2000) is a well-known action selection mechanism in cognitive science. It can account for normal action sequencing of daily routine actions in humans as well as for various errors exhibited by impaired human subjects. In this paper, we examine the potential of contention scheduling as an action selection mechanism for artificial agents, in particular robots. We first introduce the APOC architecture framework in order to summarize–in an "architecture-neutral" specification–the basic properties of the contention scheduling model. Then we analyze various aspects of contention scheduling that may cause problems in the context of the design of artificial agents and suggest modifications that may be able to overcome the difficulties, concluding that the contention scheduling model, as it stands, is not yet an appriopriate candidate for action selction in artificial agents.

Introduction

Much work in the fields of autonomous agents and robots has focussed on the problem of how to select the right action for an agent at the right time. This so-called “action selection problem” (Tyrrell 1993) has been addressed by different researchers in various contexts, depending, for example, on whether the investigation focussed on natural agents (e.g., (Lorenz 1981)) or artificial agents (e.g., (Maes 1989)). Various solutions have been proposed in the past, which can be distinguished along several dimensions, e.g., whether the action selection mechanism is *competitive* or *cooperative* (e.g., (Arkin 1998)), or whether it is *centralized* or *decentralized* (for an overview of different mechanisms, and a preliminary taxonomy, see (Pirjanian 1999)). Furthermore, explicit action-selection mechanisms could be incorporated as separate components into an agent architecture, (e.g., voting schemes as in DAMN (Rosenblatt 1997)). Alternatively, action selection can be achieved implicitly by placing constraints on the arrangement, connectivity and interactions of parts of the architecture (e.g., (Brooks 1986)).

Not all of these mechanisms, however, are equally well-suited for any task. While (motor) schema and subsumption style action-selection have proven to work well for relatively simple robotic agents, it is not clear, for example, how they would generalize beyond low-level motor control (e.g., to complex action sequences that may be realized in a large number of possible physical trajectories of a manipulator). There are various attempts in behavior-based robotics to extend reactive architectures and produce “hybrid systems” that integrate action planning into reactive control mechanisms to improve and generalize the system's motor control (e.g., (Arkin & Ali 1994)). In particular, a three-layered competitive model of action-selection, the *contention scheduling model* originally introduced by Norman and Shallice (Norman & Shallice 1980) to model human action selection and recently elaborated by Cooper and Shallice (Cooper & Shallice 2000), seems a promising candidate for action selection in natural agents. Cooper and Shallice describe a fully implemented contention scheduling model together with computer simulation experiments intended to capture the action selection mechanisms employed by humans in daily routine actions such as “making coffee”. Depending on various parameter settings, the model exhibits both behavioral patterns of normal humans and action sequencing errors that are typical of patients with a range of neurological impairments. The results suggest that for cases such as the ones studied contention scheduling can explain human behavior, and hence seems to be a robust model of action sequencing for complex action sequences (to the extent that humans perform routine actions well). It is, therefore, reasonable to ask how contention scheduling would fare as an action selection mechanism for complex, robotic agents.

In this paper, we examine the contention scheduling model proposed in (Cooper & Shallice 2000) with respect to its applicability to autonomous robots. We first introduce a general framework architecture called APOC, which will allow us to discuss various aspects of contention scheduling and compare them to other architectures. After a subsequent brief review of the main structure of the contention scheduling model, we analyze the action selection mechanism proposed by contention scheduling in greater detail and argue that contention scheduling in its present form may not be an appropriate way of structuring and designing architectures for autonomous, robotic agents.

APOC - A Framework Architecture

Why an Architecture Framework?

Analyzing different action selection mechanisms, or more generally, architectures, and comparing them can be difficult, for one since different authors tend to introduce terminological idiosyncrasies. In the best case, this may lead to insignificant misunderstandings, in the worst it will result in fundamental misconceptions. The term "reactive" as in "reactive architecture" is a case of the matter, where "reactive" can mean anything from "stateless", to "fast, timely, response", to "non-representational" or "non-deliberative". It should be clear that much theoretical weight hinges on the exact reading of "reactive" given the above differences.

Another difficulty in comparing architectures stems from the fact that authors often cannot agree on a common vocabulary, despite the fact that there may be striking similarities between what term X signifies in architecture Y and what term U signifies in architecture V. In such a case, it would be advantageous if Y and V could be viewed as belonging to a category C of components, sub-architectures or architectures, and X and U could be replaced by a more general term T referring to instances of C.

An architecture framework that supplies common categories C (of architectural features) and terms T to talk about them can overcome both problems. Hence, we will first introduce a common framework that allows us to talk about various arrangements of components and their interactions in architectures.

The Framework Components

The APOC architecture framework consists of heterogeneous computational components called "nodes", which can have any of the following four kinds of links among them: (1) *A*ctivation, (2) *P*riority, (3) *O*bserver, and (4) *C*omponent links (hence the name "APOC"). All four links can have a time parameter associated with them that determines the time it takes to pass information through the link.

Each node has a numerical activation and a priority. The former is a measure of the overall desirability for the action associated with the node, regardless of whether the action is performed directly in the environment or whether it is only internal to the agent. The latter is used to determine who gets the control over a node that is part of a collection of nodes (see the component link below). If the activation of a node exceeds a given threshold, the node is "active" and can/will perform its action.

The class of architectures that fall under this framework can all be seen to be directed multi-graphs with respect to the arrangement and connectivity of their components. The nodes could, for example, implement behaviors (understood in the sense of behavior-based roboticists like (Arkin 1998) and (Brooks 1986)), direct motor actions (such as forward movement), or actions and processes internal to the agent (e.g., reactive processes such as adjustment of control parameters, or deliberative processes such as planning).

The Activation Link The first type of link is an activation passing link, such as those used by (Tyrrell 1993) and (Maes 1989). Each node can have any number of incoming and outgoing activation links. The values on incoming activation links determine the activation of each node. How these activation link values are combined is a decision to be made on a case by case basis. For typical connectionist networks, for example, the incoming values will be multiplied by weights and summed up.

The Priority Link The second type of link is a priority link. The purpose of a priority link is to allow a node higher up in the hierarchy to change the priority of nodes further down and thus bias the system towards executing the actions associated with those nodes under certain circumstances (e.g., if a global alarm mechanism is active, as described by Sloman (Sloman & Logan 1998)).

The Observer Link The third type of link, the observer link, allows nodes to supervise the execution of another nodes. An observing node will automatically be informed about any changes (of state) that occur in the observed node (for example, whether an action associated with a node was interrupted because the node's activation dropped below threshold). Observer links remove the necessity of explicit communication for information update.

The Component Link Finally, the fourth type of link is a component link (in the sense indicated in the graphs drawn by ethologists, e.g., (Lorenz 1981)). It connects more abstract or general processes or behaviors with their constituting subprocesses or sub-behaviors. An active node N can activate or deactivate actions of nodes connected to it through component links, even if their activation is below activation threshold as long as the priority of the component nodes is lower than the priority of N (activating/deactivating an already active/inactive component node will not change anything). Optionally, a node may pass on parameters when activating component nodes (e.g., a speed may be passed to a "move forward" node).

Competitive Clusters and Winner-Takes-It-All Arbitration

The above four links provide a very general framework, in which several action selection mechanisms and behavior arbitration schemes can be analyzed. "Competitive clusters", for example, which are sets of nodes in which only the node with the highest activation can execute its associated action, can straightforwardly be implemented using an "arbitration node" with O-links and C-links to all nodes of the cluster: the arbitration node is active all the time and its corresponding action implements the "winner-takes-it-all" arbitration mechanism of a competitive cluster, where, via C-links, (1) the action associated with the currently active node is interrupted as soon as another node has a higher activation level, and (2) the action of that nodes is initiated.

Subsumption Architectures or Priority-Based Arbitration

Similarly, subsumption-style behavior arbitration can be implemented via P- and C-links, where the priorities of the nodes correspond to the layer in the subsumption architecture and P-links from upper level nodes to lower level nodes will ensure that higher nodes (with higher priorities) have precedence over lower nodes. The C-links are used to either pass parameters to lower nodes (the "inhibition and suppression links" in subsumption architectures) or to deactivate nodes (the "reset and suppression links").

Components and Links in APOC

Similar to the subsumption paradigm, the APOC framework *per se* does not define functional specifications for nodes. In contrast to the subsumption design methodology, however, where basic components are "augmented finite states machines", APOC does not specify implementation properties of basic components either, except for allowing them (1) to have an activation value and a priority, and (2) to be connected by any of the four links to other components.[1] Hence, actions and processes associated with nodes could be realized as AFSMs (as in subsumption architectures), JAVA programs, condition-action rules, fuzzy rules, etc. and nodes may implement direct motor actions, motor schemas, perceptual or memory processes, reasoning methods, etc. Furthermore, no particular arrangement of nodes is specified as this will essentially depend on the components employed at the respective nodes in the architecture.

A Brief Overview of Contention Scheduling

Basic Structure

The contention scheduling scheme, as a mechanism for action selection, forms the middle layer in a three layer architecture, in which the bottom layer is responsible for carrying out "actions" and the top layer consists of a supervisory system that monitors the progress and possibly corrects the processes in the layers below. The middle layer, is itself divided into three parts: a schema network, an object network and a resource network, each of which is hierarchically layered.

The Schema Network The schema network consists of goal directed schemas and goals. Each schema is made up of a set of several partially ordered goals, all of which have to be satisfied before the overall goal of the schema is considered to be achieved. Each goal, in turn is composed of one or more schemas, any one of which may be used to achieve the goal. A numeric activation is associated with each schema. This activation varies over time as a result of several influences that are exerted on the schema. It is the activation of each schema that ultimately leads to the selection of an action. If the activation of a schema is greater than a given threshold, then the schema is allowed to pass activation down to its component schemas (top-down influence). Other types of influence in contention scheduling come from the environment, from the schema itself, lateral from other schemas, and from random noise, all of which contribute to the activation of a schema. The environmental influence acts as a set of triggering conditions - a schema is only allowed to be active if the current conditions allow its action to proceed. An additional requirement for schema activation is that its goal has not been achieved prior to the time when it becomes eligible for activation.

The bottom of the hierarchy is composed of basic schemas, which correspond to simple actions, such as "pick-up-object". In the case of the basic schemas, an activation that is greater than the threshold leads to the execution of the associated action. Completion of this action leads to the satisfaction of the goal immediately superior to the schema in the schema/goal hierarchy. The goals for each higher-level schema are stored as part of a list in the respective schema and checked-off as each goal is achieved.

In relation to the APOC framework, a general description of the schema network is obtained through the decisions below:

1. APOC nodes are divided into two categories: schemas and goals. Goals are boolean nodes that indicate achieved/unachieved status. Schemas are sets of actions that lead to the achievement of goals.
2. Within each basic node representing a schema, A-links from the environment form a special class of inputs. These links pass a non-zero activation to the schema before it can become active.
3. Each goal node has an activation threshold of zero. Since in contention scheduling nodes do not have activations, this allows a goal node to simply pass the activation received through A-links to its component schemas.
4. Only A- and O-links are used in the network, as they most closely parallel the links described in contention scheduling. The A-link performs by definition the function of activation passing described in contention scheduling, while O-links provide a convenient mechanism for signaling that a condition has been achieved. As a result of this implementation decision, the priority of each basic unit does not affect computation.
5. The structure of the architecture with respect to A-links is hierarchical, i.e. no activation loops can be present in the schema network.
6. The basic nodes representing schemas in the bottom layer of the A-link hierarchy implement basic actions.
7. Upon completion of its action a schema node sends a signal to the corresponding goal node via an O-link, causing the goal to switch state from unachieved to achieved.

It is worth noting that in contention scheduling subschemas are not treated as subcomponents. Instead, the relation among schemas is governed through lateral inhibitory links. The following scenario is therefore feasible in the contention scheduling framework.

A high level schema is active and passes activation down to its subschemas. However, at the motor-schema level, a schema unrelated to the high level schema is highly activated by the environment and is therefore wins the competition at the motor-schema level and begins executing. This activation is a possible explanation for a number of errors exhibited by people in daily activities. In order to support this type of behavior, C-links are not used in the APOC description of contention scheduling. The use of C-links leads to direct activation of sub-behaviors, and would thus eliminate a characteristic of the contention scheduling scheme.

The Object Network Another subsystem of contention scheduling is the object network. This network parallels the schema network in many respects. Each object has an associated activation value used in determining environmental

[1] Components with no activation value are automatically taken to be active all the time. Components with no priority are taken to have 0 priority.

influence on schemas and in deciding which object to use to achieve a task when more than one applicable object is available. A different activation value is stored for each possible use of the object. In the object network, activations are affected by lateral influence, self influence, influence from schemas and random noise. The lateral, self, and schema influences are summarized in two assumptions:

> The influence of a schema's activation on that of an object representation (for a particular function) is dependent on the extent to which the object representation is employed, serving that function, in the triggering conditions of the schema. (PA 10, (Cooper & Shallice 2000) p.312)
>
> Object representations compete within functional domains. This competition is effected by a lateral influence on the activations of competing object representations, and a self influence on all object representations. (PA11, (Cooper & Shallice 2000) p.312)

The APOC description of the object network is even more concise than that of the schema network. It consists of the items below:

1. Basic nodes are nodes whose relevant information consists of a set of numeric values denoting the activation of the object represented by the node with respect to each possible use of that object.
2. Only A- and O-links are used in the network. As a result, the priority of each basic unit does not affect computation.

The Resource Network The resource network and the object network serve similar functions. The same way actions require objects in the environment to which to be applied, they also require effectors in order to be completed. A resource and a schema influence each other if the resource can be utilized by the schema. When an action is executed the most active appropriate resources are allocated to it. Basic level schemas specify restrictions on objects and resources to which they may be applied; objects and resources take the role of arguments which are filled in for each basic schema as it becomes active in accordance to the specified restrictions.

The description of the schema network in APOC is analogous to that of the object network.

Basic Parameters

In contention scheduling the activation of the various components is governed by the following several parameters:

1. *rest level activation:* the activation of a schema without input
2. *persistence (decay function):* the function that governs the return of schema activations to a rest level after the net input becomes zero
3. *random noise:* a random value added at every update to the activation to help break ties if nodes have the same activation level
4. *the balance parameters self:lateral, internal:external, and competitive:non-competitive:* specify the proportions of total activation from various sources (self-excitation vs. later inhibition, internal contribution vs. external contribution, competitive contribution vs. non-competitive contributions)
5. *activation threshold:* the number which, when exceeded by a schema activation, allows the schema to be eligible to execute its associated actions or action sequence

By definition, rest level activation, persistence, *competitive:non-competitive* ratio, and *self:lateral* ratio are the same in all three networks.

A Critical Analysis of Contention Scheduling as Action Selection Mechanism for Complex Robotic Agents

In this section we analyze some of the problems that may appear in the construction of complex robotic agents using the contention scheduling scheme. First we discuss three general problem classes: (1) problems arising from interruptions and failures of schemas, (2) problems resulting from modifications of the architecture, and (3) problems that result of interactions with the environment. Then we focus on implementation-specific issues in the design of complex agents. It should be noted that some of these problems (such as interrupt handling) are common to AI planners and not necessarily restricted to contention scheduling

Three General Problem Areas of Contention Scheduling

Contention scheduling as action selection mechanism has various strengths, including the ability to use different methods for the accomplishment of the same goal. The potential of CS as illustrated by Norman and Shallice through their coffee making example, however, is given in a very restricted, simple context, in which the amount of activity interference of different schemas is small. By looking at a larger, more realistic systems, several issues come to light, which we will discuss in this section.

Interrupted Actions and Failures of Actions to Complete A core assumption of contention scheduling states that influences on a schema activation come from five sources: top-down influence, environmental influence, self influence, lateral influence, and random noise. This assumption excludes bottom-up feedback from component schemas to their superordinate schema. Such feedback, however, may be very practical, if not required to be able to cope with interrupted actions and/or failure of actions to complete at the basic schema level.

First note that while a basic schema node whose action has a direct effect on the environment can use properly directed environmental queues to infer the failure and or interruption of its associated action, nodes further up in the schema network can at best receive information about the failure from the supervisory layer (in terms of excitation and/or inhibition of their activation!). This mechanism is by itself both slow and prone to failure. Consider the example in Figure 1 (taken from Norman and Shallice).

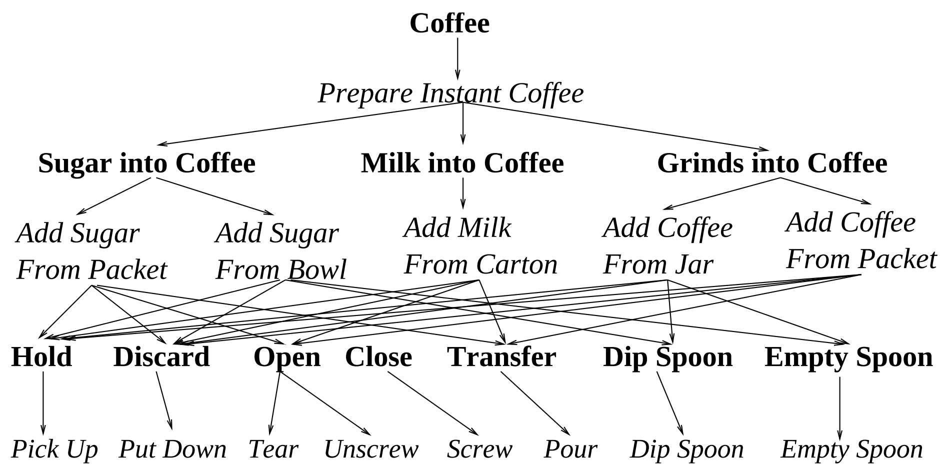

Figure 1: Coffee making example

Two problems connected to potential failures of actions can be seen from this structure. First, the "Close" goal is unconnected to any other node, hence recovery if the "Screw" action fails (i.e., the goal is marked "achieved" regardless of whether the lid is really on after finishing the action, for example) depends on the success of operations further removed from the action itself. Hence, the likelihood of not detecting the failure of the "Screw" action (e.g., by the supervisory system) is increased. The second problem arises when a "Pour" action fails (e.g., because the milk is spilled). Such a failure would, under the current design, allow the rest of the coffee preparation process to proceed even though the overall goal is no longer feasible.

Furthermore, some actions may require a certain time interval for completion (because their effects are otherwise meaningless or inappropriate, e.g., stirring the coffee for two hours). However, there seems to be no mechanism (even at the level of simple, motor schemas) in contention scheduling that oversees and possibly facilitates the successful completion of actions and can correct failures by interrupting, reinitiating, or aborting the same or other actions (except possibly for the supervisory system, but then the way in which this system can intervene needs to be specified).

As mentioned above, interruptions of basic actions may pose another problem for contention scheduling. While the unorderly state the system may be in after a basic action has been interrupted is considered a virtue of contention scheduling *qua* model of human action selection, such an undefined state is not necessarily a desirable feature for artificial agents (e.g., if the agent is in control of an aircraft, where failures of basic maneuvers can have devastating consequences if not countered quickly and appropriately).

Finally, another problem caused by interruptions may be connected to "peripheral assumption 3":

> " When a schema's state changes from selected to unselected, all subgoals on the schema's subgoal list are marked as unachieved. ((Cooper & Shallice 2000) p. 310)

If a high level schema like "make coffee" loses its activation and gets deselected, then all its goals will be marked as unachieved, including, for example, already achieved goals like "add sugar". When the schema gets activated again, it will cause sugar to be poured in the mug again, although sugar is already in the mug (assuming this cannot be directly observed). Again, while this may be appropriate for a model of human action selection, its seems that agents should be able to remember completed actions, not only because it reduces overhead, but also because it can prevent undesirable results that may result from performing the same action twice or more times.

Learning New Actions Contention scheduling does not deal *per se* with modifications of any of the three subnetworks (e.g., the object, resource, and schema networks). However, humans (as well as complex artificial agents) need to be able to learn new complex actions, i.e., action sequences (in addition to learning the low level motor control that may be required for some higher level action, e.g., grasping a mug). What is unclear then is how such a hierarchy could be learned and how an existing hierarchy of schemas could be modified or augmented. It seems that adding new nodes to the schema network, for example, would impact the relative times it takes to activate the respective schemas (e.g., by adding another schema to a group of already competing schemas, lateral inhibition will be stronger on for the schemas than it was without the added schema; but this means that it could take longer for schemas to reach activation threshold). The only way to get around this problem would be to change the global "competitive:non-competitive" parameter. But then it is not clear exactly how it should be adjusted.

Further problems are connected to questions of whether a new schema can be executed in parallel. Suppose the system only learns one way of performing the high-level action associated with a high-level schema S, which puts S in competition with other schemas, since they all share the same low-level action. Upon encountering alternatives, however, S would not have to be in competition with other schemas if these alternative low-level actions can be performed in parallel. Yet, it is not clear how the schema network would have to be restructured in this case. Similar problems arise if the resource or object networks need to be modified.

Finally, variations in accomplishing a task may pose another problem, given that contention scheduling intrinsically relies on a fixed break-down of actions into sequences without allowing to "parameterize" individual actions. The process of coffee preparation, for example, is not always exactly the same. The amount of added sugar and milk, the strength and temperature of the coffee, etc. may all vary (e.g., depending on mood, time of the day, etc.). Without the possibility to parameterize low-level actions, different schema-trees would have to be added, e.g., for "making strong coffee", making "coffee with milk", etc. and, of course, changes would perpetuate up the schema hierarchy (e.g., up the "make breakfast" hierarchy). In addition, new inputs would have to be allowed to the schema network (e.g., inputs from a "mood" system) to be able to select meaningfully among the different alternatives.

Interactions with the Environment One of the strengths of contention scheduling is its ability to integrate Gibsonean affordances (Gibson 1979) with action selection, i.e., whatever object is present in the environment and perceived, will feed activation to the schema that represents an action involving this object. However, there may be problems with such a direct coupling of perception and action in that an action could be triggered by mere presence of an abundance of instances of one and the same object type without there being any additional reason as to why the action should be perform (e.g., suppose you are on a parking lot and perceive hundreds of cars, then the affordance of "driving a car" may be so strong as to make you actually drive it without any further reason).

Another problem with perceived objects is connected to the above-mentioned issue of variations in routine actions: some objects or tools can be switched for others if they agree with respect to the functionality required for an action. For example, it would be possible to use a table spoon or the tip of a kitchen knife to scoop sugar in the "coffee making"-task, if no tea spoon were present but one or both of the other objects were. However, what action an object affords needs to be explicitly coded in the schema network. Hence, there are problems of generalization and of scaling, as it is almost impossible to add all the potential applications (i.e., "causal functions") of even simple objects that may be relevant to a given action sequence.

Finally, there does not seem to be any provision in contention scheduling to prevent a low level schema from being active (e.g., because of environmental stimulation) even if its source schema is not, which could lead to interference among various actions.

Design Issues for Complex Robotic Systems

While any of the previously mentioned issues may present a problem for the (complex) artificial (and possibly natural) agents, in this section we will focus in particular on problems that may arise in the context of designing complex artificial agents.

Complex Goals One issue in the design of complex agents is the representation of overall goals of the agent. In contention scheduling, goals are broken down into subgoals that are specific to each schema, which each keep track of their own goal list. The authors

> " assume that, within contention scheduling, a schema's goals are "ticked off" as they are achieved by the system. ((Cooper & Shallice 2000) p. 309)

In many instances this is an efficient way of dealing with complex goal achievement. Yet, if the number of goals of a schema is not fixed or given in advance, this approach could lead to severe implementation and scalability problems. Consider a robot equipped with sonar sensors, which is supposed to map a room. It does not have any prior information about the layout, nor can it obtain (being limited by the nature of its sensors) any general information about the room (such as the number of walls). Figure 2 depicts the schema-goal structure.

For such an agent, it is not clear how a contention scheduling action selection scheme could be effectively implemented, as contention scheduling requires that all subgoals of a schema be achieved before a complex schema can accomplish its action.

Basically, there are two possible implementations within the contention scheduling paradigm for dealing with a variable number of subgoals. The first, adding a different schema for each possible number of goals, does not seem feasible (it could lead to an explosion of the number of schemas beyond what is practically feasible). The second involves implementing the process of mapping the room as a process that continues until a condition is satisfied. However, the contention scheduling architecture cannot deal well with perpetual conditions or goals that consist in maintaining a particular state (possibly until an interruption is encountered, such as "maintaining the distance to the car in front of you on the highway"). The problem lies in the way goals are represented and checked for achievement, which does not work easily for perpetual or ongoing goals. Rather, such goals and their associated actions would have to be represented outside the schema network. But then separating action sequences that can be completed in a timely manner from sequences that are in some sense repetitive becomes an issue, which leads to all kinds of questions about how longer-term goals should be represented, etc.

A related issue is that of more abstract schemas, which may have goals that are either not well-defined or that change with time. A system could, for example, notice that attempts to scoop coffee out of a container repeatedly fail and start experimenting with different methods of getting the coffee out (e.g., turning the container upside down), and decide that since this method works better in general to use it from now on, where the amount of coffee is now determined in terms of pouring time and pouring angle instead of the number of scoops.

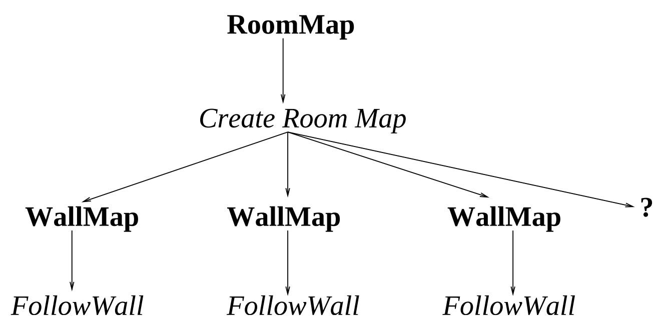

Figure 2: Map making example

Reflexes Fast, reflex-like motor actions, triggered by various environmental conditions, are of great importance in robotic agents (e.g., to prevent damage, improve performance of subsequent actions, etc.). Hence, an action selection scheme needs to be able to incorporate such reflex mechanisms that can interrupt ongoing behaviors, perform whatever action is necessary, and return control to the interrupted behavior.

In contention scheduling, interruptions of actions can only take place when a schema's activation drops below the predetermined activation threshold (e.g., because another schema's activation is rising). Hence, reflexes would have to use this activation passing mechanism to obtain control to execute their associated actions. For example, they could be implemented as bottom level action schemas whose only activation comes from the environment. However, they would have to be in competition with every low-level action schema that deals with the same effectors as the reflex, since they need to be able to interrupt it if necessary. This raises a question about the inhibitory influence: since inhibition is the same everywhere in the network, it may not be possible for the reflex scheme to become excited above the threshold level for activation fast enough if there are too many other schemas in competition with it. Yet, for it to be a reflex and serve that function, timely activation has to be guaranteed, especially in real-world domains in which robotic agents operate.

Furthermore, once a schema is interrupted by a reflex, it is not clear whether that schema will become active again and can continue its action once the reflex stops. For one, other schemas may become active, and it may also not be possible to continue the action. For example, interrupting a pouring action (of sugar into the mug) may lead to a state where the system does not know how much sugar has been added to the coffee. If the right sugar content of the coffee is of great importance, the best bet for the system is then to start over again. However, there is no mechanism in contention scheduling to recognize failures and interruptions that could provide this kind of feedback.

Discussion

Contention scheduling is a very powerful action selection mechanism. It has great biological plausibility as a model of human routine action selection and execution and seems to have great potential for applications in artificial agents.[2] Some have even argued (e.g., (Glasspool 2000)) that the contention scheduling model and other action-selection mechanisms from artificial intelligence and robotics (e.g., (Maes 1989)) are converging, hence pointing to success of such a three-layered architecture for action-selection. We have, however, discussed several issues of contention that may cause problems with such a three-layered architecture if applied to artificial agents, in particular, robotic agents.

The first set of potential problems involved interrupted actions and failure of actions to complete, which are partly due to the uni-directional flow of information in the architecture. Extending the architecture to include a feedback mechanism from component schemas to their parent schemas would allow for the construction of a localized failure/interrupt handling mechanism. The feedback could be implemented using either A- or O-links, and would comprise information about the execution status of the subschema that the parent schema can use to determine in case of interrupts or failures which action to perform next. It is straightforward to extend the current model by such a mechanism.

Furthermore, given that a parameter passing mechanism is already implemented, which is used to select appropriate resources to apply to objects for a given schema (e.g., "use the right hand to pick up the mug"), this mechanism could be easily generalized to allow other parameters to be passed to schemas. That way parent schemas could specify variations in the execution of their component schemas, thus solving the variation problem.

The problems related to goal representations, however, are more difficult to remedy, as they involve the lists of goals that each schema needed to determine the completion of its activity. As it stands, this mechanism for checking goal achievement cannot be easily modified to overcome the problems mentioned above. Rather, it would be better to implement a different mechanism, which signals the completion of the action associated with a schema if a series of conditions are met, which do not necessarily have to be subschema goals. For example, the map making schema can

[2](J. Garforth & McHale 2001) implemented contention scheduling in a robotic agent. Unfortunately, the architecture is not specified sufficiently and the results of the experiments are not reported in enough detail to be able to assess the success of this application.

require that at the end of the mapping process the agent be within a tolerable error from the point where it started the process. As long as that condition is not satisfied, the “follow wall” subschema is kept active.

A second problem related to goal representation and goal handling involves persistent goals. With persistent goals, the aim is to maintain one or more conditions over a (possibly long) period of time. Consider the task of driving along a highway. One of the goals that make up that task is that of maintaining a relatively constant speed - assuming a normal flow of traffic. This goal can involve pressing the accelerator pedal (on an upward incline) or the break pedal (on a downward incline). It is also a persistent goal, as it involves the maintenance of a condition, rather than the achievement of one.

Achievement of such goals should hence be measured in terms of the presence or absence of the conditions they maintain. Such goals could be better realized in an implementation, in which they can signal *failure* (e.g., through the use of O-links) to the schema immediately superior in the hierarchy. The execution of that schema would then proceed normally as long as the necessary conditions for the persistent goals are met.

Interaction between the environment and a contention scheduling-based architecture has also been shown to have several potential problems, such as excessive excitation of a schema (by virtue of “affordances”) in the absence of an active super schema. A possible solution would be to allow unselected schemas to exert a negative influence on their successors in the architecture, thus making it more difficult for them to become active by environmental conditions alone.

In sum, we believe that contention scheduling is a very interesting candidate as action selection scheme for complex agents. However, in its present form there are several issues, crucial to the proper functioning of artificial agents, that are not addressed well or addressed at all by the contention scheduling architecture. It may be possible to extend the contention scheduling model in ways that we indicated above to overcome some of these difficulties, and we are currently in the process of implementing such an extension on a robot to test its viability and effectiveness. We expect our results to conform the theoretical analysis above, in which case extended versions of contention scheduling may prove applicable in robots. As it stands, however, we take contention scheduling to be more of a theoretical model of action selection than a practical design that can be easily and reliably applied in artificial agents.

References

Arkin, R. C., and Ali, K. S. 1994. Integration of reactive and telerobotic control in multi-agent robotic systems. In *Proceedings of the Third International Conference on Simulation of Adaptive Behavior*, 473–478.

Arkin, R. C. 1998. *Behavior-Based Robotics*. Cambridge, Massachusetts: MIT Press.

Brooks, R. 1986. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation* RA-2.

Cooper, R., and Shallice, T. 2000. Contention scheduling and the control of routine activities. *Cognitive Neuropsychology* 17(4):298–338.

Gibson, J. 1979. *The Ecological Approach to Visual Perception*. Houghton Mifflin, Boston.

Glasspool, D. 2000. The integration and control of behaviour: Insights from neuroscience and ai. Symposium: How to design a functional mind.

J. Garforth, A. M., and McHale, S. 2001. Executive attentional control in autonomous robotic agents. In *Proceedings of the 2001 International Conference on Intelligent Agent Technology*, 479–483.

Lorenz, K. 1981. *The Foundations of Ethology*. Springer-Verlag, New York.

Maes, P. 1989. How to do the right thing. *Connection Science Journal* 1:291–323.

Norman, D., and Shallice, T. 1980. Attention to action: Willed and automatic control of behaviour. Technical report, University of California, San Diego, CA.

Pirjanian, P. 1999. Behavior coordination mechanisms - state-of-the-art. Technical Report IRIS-99-375, Institute for Robotics and Intelligent Systems, School of Engineering, University of Southern California.

Rosenblatt, J. 1997. *DAMN: A Distributed Architecture for Mobile Navigation*. Ph.D. Dissertation, Carnegie Mellon University Robotics Institute, Pittsburgh, PA.

Sloman, A., and Logan, B. 1998. Architectures and tools for human-like agents. In Ritter, F., and Young, R. M., eds., *Proceedings of the 2nd European Conference on Cognitive Modelling*, 58–65. Nottingham: Nottingham University Press.

Tyrrell, T. 1993. *Computational Mechanisms for Action Selection*. Ph.D. Dissertation, University of Edinburgh.