AIAA-94-1291-CP

PLANNING IN SUBSUMPTION ARCHITECTURES

Eugene C. Chalfant

University of Southern California
Los Angeles, California
echalfan@pollux.usc.edu

Abstract—A robotic planning and control system based on the subsumption architecture is described. The subsumption planner extends the purely reactive subsumption architecture by extending the sensor space (from which the behavior modules are triggered) into a virtual future, and augmenting the behavior module with a cause-effect predictor triggered by the same sensory situation as the reactor. Virtual sensor space is used by the planner as a scratchpad to visualize alternate plans. The predictor contains a partial world model relevant to its particular behavioral expertise. The collective network of predictors operates in parallel with the reactive network forming a recurrent network which generates plans as a hierarchy. Details of a plan segment are generated only when its execution is imminent according to the principle of least commitment. An implementation of subsumption using object oriented design is proposed. The behavior of the subsumption planner is demonstrated in a simple maze navigation example. The subsumption planner is expected to improve the robot's performance by reducing feedback delays and unnecessary detours. It provides a framework for general behavioral planning in real-world robots of the subsumption style.

1. Introduction

Reactive robotic control systems, such as the subsumption architecture, have enjoyed popularity among the research community for the last few years. Robots built according to these principles are very successful at performing tasks in unstructured, real-world environments. However, reactive systems tend to behave in a pre-programmed, rote manner. Selecting alternate courses of action based on previous experience or reasoning (i.e., deliberative behavior) must be designed into the behavior itself rather than being an emergent property of the network interconnectivity. Recent work has recognized a need to incorporate deliberative planning capabilities in real-world systems.

Planning is essentially the ability to look ahead and predict outcomes of actions (or inactions), and to make decisions based on that knowledge. A plan is a sequence of decisions to be made in the future. These decisions assume a particular expected sequence of states. Anticipation of future states gives the planning robot advantages in efficiency (via the ability to avoid known dead ends), in flexibility (replanning on the fly), and in reaction speed (eliminating most of the delay inherent to reactive feedback-only systems). Autonomy in the real world demands predictive and deliberative behavior in addition to a reactive component.

The subsumption planner uses a parallel distributed computational paradigm based on the subsumption architecture for the control of real-world-capable robots. This approach is derived from a number of various disciplines including ethology, the theory of animats, and computational neuroscience. Plans are represented as trajectories in a time-extended virtual sensor state space. States along this trajectory represent the future as the robot expects it to appear, in terms of its own senses. Virtual sensor state space is used as a planning tool to visualize the robot's anticipated effect on its environment.

Decision sequences are generated by the planner based on the environmental situation expected at the time the robot must commit to the decision by acting on it. Between these decision points, the robot performs in a pre-programmed manner, limiting its reactions to avoiding obstacles, stalls, and danger. A rudimentary, domain-specific partial world model contains enough information to extrapolate the end results of rote behavior between decision points, and thus to predict the sensed situation at the next decision point.

By constructing plans with little detail as long as they are far in the future and filling in details only as their execution becomes imminent (the principle of least commitment), failed plans can be discarded without much resource cost. Rough plans are represented as indefinite trajectories between a few well-defined waypoints in the state space. These waypoints are the expectations generated by the predictors. Initially, they are spread far apart in time—the plan is coarsely resolved. Rough planning consumes few resources. As the first plan segment comes closer to fruition, details are added just in time for execution. These multi-resolution plans are built from the outside in.

Planners need a world model to evaluate alternatives. One objective of the subsumption planner is to distribute the world model in a plausible way across the entire decision-making system. Each behavior module contains a discrete piece of expertise. These expert agents contain not only action generation routines, but also cause-effect predictors. The planner is implemented as a network structure which parallels the behavior module network of the subsumption architecture.

Implementation of subsumption as an object-oriented design simplifies the implementation of a software simulation, as well as providing organizational and developmental flexibility. The subsumption network structure lends itself to the use of distributed hierarchical encapsulating objects. In addition, the hierarchy enables prioritization of active behaviors to occur before the generation of actual motor outputs. This simplifies the prediction mechanism.

2. Related Work

As the shortcomings of monolithic, centralized robotic controllers and planners became obvious, new distributed architectures were proposed. These were rooted in a variety of different nontraditional disciplines, including ethology, biology, neuroscience, physics, psychology, and sociology. A strongly ethological and intuitively elegant approach is espoused by Albus[1]. Marvin Minsky, a pioneer in artificial intelligence, theorizes about the nature of information processing in the human mind, speculating that thought is composed of the collective actions of a society of individual, subintelligent agents acting cooperatively[2]. The new architectures share many features with object-oriented paradigms. Schema theory[3], in particular, models intelligent systems from the perspective of brain theory as hierarchical networks of nested object-like schemas. Schema theory and object-oriented design are both rooted firmly in distributed artificial intelligence.

Sensor and motor information (indeed, any type of information at all) can be described in terms of dynamic high- or infinite-dimensional state spaces[4]. A trajectory through this state space represents the time evolution of information or concepts. This generic framework has been developed into a description of physical and mental behavior, which can describe many diverse information processing techniques and knowledge representations.

The subsumption architecture developed by Brooks at MIT[5, 6, 7] introduced a new perspective on building robots for the real-world, often called creatures or animats. Connell[8] further develops the subsumption technique, describing an autonomous robot whose job it is to find and collect soda cans in a lab and deposit them in a receptacle. This robot has served as a testbed for an in-depth study of subsumption and its numerous advantages over earlier methods for controlling real-world robots[9]. The major achievement of the subsumption architecture is to demonstrate how to combine many isolated pieces of robot control into a single, working real-world machine operating under a single, coherent, and consistent architecture.

Earlier robots were effective only in toy worlds: well-defined environments, such as in manufacturing, where repetitive movements could be guaranteed to be successful. These applications do not require much sensory ability. The robots built by Brooks react in real-time to changes in the environment. They rely on a rich suite of sensors to provide information about the environment on which to act. The actions themselves are generated by a network of behavioral modules whose outputs are mediated by a hardwired arbitration network of gating nodes. This network gives certain higher-level behavioral modules responsibility for, and control over, the outputs from lower-level modules.

In addition to performance advantages due to direct implementation on parallel hardware, the subsumption architecture provides all the benefits of a distributed system, including scalability, graceful degradation, robustness, simpler elements, and biological plausibility. Much of the intelligence of the system emerges from, and is embodied in, the network connectivity.

The lack of an explicit world model was originally seen as an advantage of subsumption. Using the real world as its own model solved the problems of keeping the model up to date and deciding what was important to include, as well as what form of representation to use. However, without a world model, a system or organism cannot anticipate the effect of an action until that effect is actually sensed (giving rise to feedback delays). This can lead to poor reaction time and potentially life-threatening slowness.

The purpose of a world model is twofold: It keeps track of the state of the world, and it describes cause-effect relationships due to actions (i.e., moving an object) or inactions (allowing an object to fall). These are data structures and processes respectively. The cause-effect relations may be due to laws of physics, to the robot's own actions, or to the actions of another agent, in order of increasing difficulty of prediction. In the subsumption architecture, these cause-effect relations are implicitly designed into the behavior modules. For example, termination conditions assume a cause-effect relationship due to physical laws. The result of actively moving an object from x to y is that the object eventually exists at y. The fact that the object is at y can either be represented internally in a world model, or, in the subsumption model, sensed again only when necessary.

Recently, several modifications and extensions to the subsumption architecture have been proposed. These address limitations or inefficiencies of subsumption such as the lack of a world model, or its lack of learning ability.

Mataric[10] extends subsumption by implementing knowledge representation as a map (a kind of world model) for navigation by integrating an internal landmark-based map representation built as behavior modules. Behavior modules are used here to *represent* as well as *act*. In this case, each module represents a landmark. The building blocks of the subsumption architecture remain fundamentally the same; Mataric describes a new usage of them.

Dorigo and Schnepf[11] propose a learning technique for behavior-based robots in which new behaviors are created and added to the robot's repertoire using a genetic algorithm. The behavior-based architecture is based on ethological theories developed over the last eighty years[12]. They note that behaviors have been constructed which are well tailored to specific tasks, but no conceptual model exists for relating behaviors to each other. Learning is achieved by evaluating new behaviors and including them if they perform well. The resulting system is a synthesis of behavioral and genetics-based paradigms for intelligent real-world behavior.

Learning by progressive modification of a repertoire of reactive behaviors is described by Lyons[13], using a process-algebra language. The concept of a planner as a separate subsystem which interacts with a reactor is presented. The plan is contained in the reactor as a set of processes. The reactor adds and deletes processes, tuning the reactor to perform the task more robustly over time. The planner has a repertoire of plan elements and the knowledge of how to fix shortcomings in the reactors overall behavior. It provides the system with a global perspective on task execution which is lacking in the purely reactive system.

Traditional planning systems constructed plans off-line, often in a separate centralized planning subsystem. Planners were seen as logical engines which generated a complete, detailed plan as a result of a search through a tree of possible action sequences[14, 15]. These plans worked best in a toy world. Distributed planning systems were eventually developed as a consequence of the focus on distributed artificial intelligence in the late 1980's. One of these was the Distributed Vehicle Monitoring Testbed[16, 17, 18] which distributed partial plans among independent agents. In this scheme, plans were subdivided spatially.

In the past, planning has been studied in isolation. It is now becoming widely accepted that the predictive, deliberative planning component is necessary to complement the reactive behavior-based system. These two systems must be highly integrated, yet distributed across the network. Planning, in a sense, represents the antithesis of reality-based reactive systems. A planner must imagine nonexistent realities and visualize and evaluate alternative future actions in that virtual reality.

3. Architecture

Reactive robot controllers implement a feedback loop for all activity (Fig. 1a). Feed*forward* loops (Fig. 1b) are more responsive and exhibit smoother, more desirable control, provided the model is accurate. The feedforward control technique has been used to mimic the smooth, precisely controlled behavior of the human arm by a robot arm using highly non-linear air-bladder "muscles" [19, 20, 21]. The feedforward loop relies on an internal model of physical cause and effect to predict and generate appropriate actions in real time, before their effect can be sensed. The Kawato-Katayama arm trajectory generators, implemented as neural networks, act as continuous-time predictive models the physics of arm behavior.

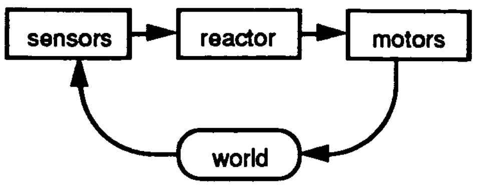

Fig. 1a. Feedback only reactive system

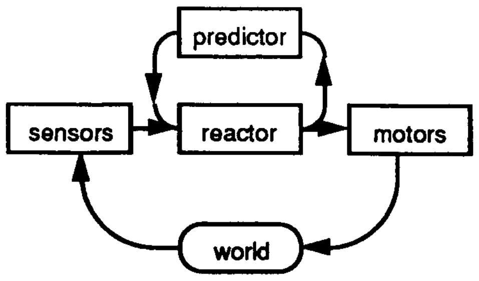

Fig. 1b. Feedforward reactive/predictive system

The subsumption planner adds to traditional subsumption the ability to predict the effect of robot actions. The action-generation expertise is collocated with the knowledge of cause-effect relationships in the behavior module associated with that domain. This cause-effect knowledge is encapsulated in a predictor which anticipates the expected state resulting from performing the behavior. The expectations are represented as state vectors in the virtual sensor state space representing future states—the collection of these states together define a trajectory representing a future course of action.

Subsumption networks are often seen in which information flows in one direction from input to output. There are no recurrent links in this type of network. The lack of recurrent links allows only simple reactive behavior to be produced, precluding complex dynamic behavior. In the same way, non-recurrent neural networks such as the back propagation network do not contain any recurrent information flow. Non-recurrent networks are not dynamical systems, and therefore they cannot exhibit chaotic or even oscillatory time-varying behavior. This severely limits the behavioral complexity and sophistication of these networks (although they are more easily understood). Similarly, in the subsumption architecture, any complex processing occurs *within* the behavior modules, rather than between them. (Some specialized subsumption networks have been designed with recurrent links; those for the control of walking, for example.)

The subsumption planner allows for three general forms of information flow. The sensor subsystem contains an afferent abstracting flow, and the motor subsystem contains a de-abstracting efferent flow. The subsystem between these two, comprising the decision making mechanism, is constructed as a network which allows a great number of information flows in both forward and backward directions. The forward flows are essentially identical to those of the conventional subsumption architecture and represent reactive behavior such as tracking or avoidance. The backward flows create loops in the information flow; these recurrent flows generate the visualization of future actions and planning. These recurrent flows are model generated predictions which are fed *forward* (i.e., in the same direction as information in the real world—from effectors to sensors) toward the sensory subsystem and treated as imaginary, virtual sensory situations. The key point is that the internal modeled information flow mimics the outside world's physical cause-effect information flow. Multiple iterations of this loop propagate predictions farther into the future, although any modeling errors will accumulate. These sensory situations are ordered in time and represent plans for future behavior.

In the traditional subsumption architecture, the responsibility for reacting to the environment is decomposed and delegated to a number of reactive behavior modules. A behavior module can be seen as an agent which is an expert in one discrete domain of behavior. These modules in turn may have hardwired managerial control over an entire network of subordinate behavior modules. The control is manifested as subsumption, whereby the manager commandeers control over the robot whenever it wants. The manager handles special situations only; the common situations are ignored by the manager and handled by its subordinates. If the specific triggering situation for the manager does not exist, some subordinate, more generalized behavior probably *has* been triggered by a less specific sensory situation. At the lowest level, a behavior may be continuously triggered by default; in effect, it is triggered by *any* sensory situation.

4. Extensions to Subsumption

In order to implement the subsumption planner, four extensions to the traditional subsumption architecture are needed:

- Sensor and motor space abstraction
- Virtual future sensor space
- Cause-effect predictors: augmentation of behavior modules

- Hierarchical organization: object-oriented implementation of arbitration

Sensor and motor space abstractions extend the real-world interface (input/output) information spaces, allowing for more sophisticated, abstract triggering and action-generating mechanisms for the behavior modules. Virtual future sensor space provides a scratchpad for construction of plans. Cause-effect predictors are associated with the behavior modules and generate the plans themselves as trajectories in the virtual sensor space. The hierarchical organization is an alternate object-oriented implementation of the subsumption architecture.

4.1 Sensor and Motor Space Abstraction

Sensor space is a very-high-dimensionality representation of the information entering the robot from its external sensors. After processing and combining raw sensor data, abstract sensor data containing composite information in a more easily assimilated form, or a form with higher information density, can be created. These abstractions have been called "logical sensors"[22]. They are generated through the fusion of raw sensor data. Examples in robot vision include edge detectors, novelty detectors, motion detectors, or highly abstract sensors such as face recognizers. Cross-modal abstractions sense location or orientation using a number of sensor sources to reduce uncertainty. A logical sensor might determine location by combining information gleaned from a number of raw sensors, such as sonar, visual clues, radio landmarks, etc.

In the original subsumption architecture, behaviors are triggered directly from raw sensor inputs. The entire extended sensor space including raw sensor data and abstract, derived sensor data, is available to trigger the subsumption planner's behavior modules. Any particular behavior module is sensitive only to a small localized subset of this sensor space, similar in some measure, for example a small visual patch. The sensitivity of a behavior module is also localized in level of abstraction—a simple, low-level behavior may trigger on contact with a single touch sensor; a more abstract, higher-level managerial behavior on recognition of a complex object such as a familiar face.

The abstract sensors are derived from the raw sensors; the raw sensor space by itself is complete. The abstract sensor space extension is a conceptual device to provide behavior modules with a common pool of sensed information from which to trigger. A behavior module, rather than triggering on a complex pattern which could potentially be multi-sourced and multi-modal (as well as uncertain and noisy) in raw sensory space, may trigger on a simple abstract pattern, or even a single highly processed attribute in abstract space.

These high dimensional spaces are conceptual constructs. Obviously, to implement them faithfully following the theory is extremely wasteful and probably impossible. The implementation can be drastically optimized while still following theory. For example, to conserve resources, the abstraction of raw sensors and data fusion should only be performed by request.

Complex motor actions generated by the motor subsystem are also abstracted into motor pattern generators (called "central pattern generators" by neuroscientists). Locomotion is a repetitive abstract motor pattern which is comprised of a number of individual motor actions. The relatively complex repetitive pattern of actions are generated by a single manager behavior which directs simpler, lower-level behaviors to generate the rudimentary component actions. Variations of locomotion, such as speed, gait, direction, can be supplied to the abstract locomotion behavior as parameters.

4.2 Virtual Future Sensor Space

Sensor space, extended in dimensionality by the incorporation of abstract sensor dimensions, is also extended orthogonally to represent a time dimension—the extension of the sensed environment into the future. This extension will be used to provide a working scratchpad for plan generation.

This extension provides a conceptual framework for the construction of plans. A plan is a visualization of a sequence of events. The most complete, natural, and efficient way (for that matter, the only possible way) to represent a visualization is in terms of the same modality in which it will actually be sensed. Since the extended sensor space contains all possible sensory situations, the visualization of "how it will appear to the senses" is also contained. The difference between real-time sensory situations and virtual sensory situations is in completeness (everything need not be represented), resolution (irrelevant details may be omitted), and uncertainty (multiple possible values or fuzziness).

The plans generated by the predictors are represented as a sequence of nodes or waypoints, relatively firmly located in state space, at distinct future times, connected by indefinite links. As the plans mature or become more imminent (the plans are also refined by the predictors), they become more firm in location, and intermediate waypoints emerge.

Multiple alternative plans are tagged with plan quality measures. The plans are evaluated by the predictor. When a choice of plan is necessary (i.e., when a complete replan is necessary, or upon embarking on a new task), the highest quality plan is selected. The robot will then begin executing this plan.

4.3 Cause-Effect Predictors

Predictors are agents associated with behavior modules. The original subsumption style action-generating behavior module, which we now call a *reactor*, is triggered by a sensory situation in the current sensor subspace. By augmenting the behavior module with a predictive mechanism, called the *predictor*, the effect of the reactor on the environment can be predicted. The predictor is triggered by the same sensory situation as the reactor in the current *or any future* sensory subspace. The predictor then generates a trajectory from the trigger point to the future point representing the predicted resultant effect of performing the behavior.

Predictors are not required for low-level, reflexive, or protective behaviors such as obstacle avoidance or tracking. These behaviors avoid anomalous states, such as being stuck in a corner; they guarantee that the robot remains in a nominal space. Higher level behaviors can expect that the anomalous states will be avoided. The lowest level behaviors are purely reactive. This reliance on low-level behavioral guarantees make the prediction job of the higher level behaviors easier. In general, predictors are necessary only for manager behaviors.

The predictor is insensitive to its trigger source—whether it is actually being sensed at the present moment, or if the sensory situation is an imaginary construction of some previously triggered predictor. The idea is to visualize a sequence of events using exactly the same sensory computational pathway as would be stimulated during the actual performance of the sequence. The differences are that: 1) the actual raw sensor transducers are not stimulated, 2) *conceptually*, the behaviors are displaced along the time dimension of the sensor space, and 3) the reactors do not generate motor commands. The same mechanisms used for reactive behavior (the trigger mechanisms and the arbitration of actions) are used for planning.

Obviously, the prediction is defined only in the subspace in which the reactor may potentially have a repeatable, predictable effect—a reactor which closes the robot's gripper will, in general, have no predictable, correlatable effect on sensed ambient light intensity.

As in schema theory, a subsumption planner predictor is a black box. The internal mechanism of the predictor is not important to the functioning of the entire distributed system; only the externally observable functional behavior (i.e., its role in the architecture) must be well defined. The implementation may use a procedural algorithm, a neural network, specialized hardware, or any other technique to perform its function.

4.4 Hierarchical Organization

While planning, no outputs should be generated from the activity of the behavior modules. The reactor does not generate output if the trigger is in the virtual sensor space. However, since the subsumption architecture's arbitration network (the collection of output wires along with their connectors) are connected to reactors which generate outputs whenever they are triggered, it is difficult to determine the actual controlling behavior module without actually generating motor outputs to send through the network. This would require some sort of action gating mechanism at the output of the arbitration network to inhibit the passage of motor commands. The result of the arbitration needs to find its way back to the predictor also.

The connectivity of the subsumption network defines its global behavior. Behavioral modules which are triggered upon sensing special environmental situations subsume lower level behaviors (if the subsuming behavior was not triggered by a special case of the subsumed behavior, the subsumed behavior would not be triggered in the first place, and the subsumption relationship would be inappropriate, i.e., in a properly designed network the subsumed behavior should already be triggered and operating when the subsuming behavior takes over). In a sense, the manager behavior commandeers control of the entire system because it believes it is best suited (in the eyes of the designer) to deal with the problem at hand.

The triggering of a module does not automatically cause the robot to perform the actions generated by this module. Several behavior modules may be triggered by the same sensory state. The actions must be mediated by the subsumption or arbitration network. This network determines which behavior may assume control of the entire system at any one time. In general, behaviors which are triggered by a smaller, more exclusive region within the sensory space will be regarded as more appropriate than more generic behaviors. The result is that behaviors which trigger upon special situations, or exceptions to the general rule, will have priority over other behaviors. These more sophisticated, specialized manager behaviors "subsume" lower-level, or simpler behaviors.

In order to retain "ownership" of the motor commands by the generating behavior, an alternative is to perform the arbitration before the commands are output by the reactors. By doing this, the behavior module knows that it has been selected (on the basis of either real or virtual senses) as the controller of the entire system, and the predictor can act accordingly. If the trigger source is the virtual sensor space, the arbitration still occurs, and the result is known locally to the behavior module.

The subsumption architecture triggers all eligible (even inappropriate) behavior modules to generate actions in parallel. By giving each module some awareness of its own place in the control hierarchy, and some visibility into the activity of behaviors which may override its own eligibility, the number of eligible, active modules may be reduced to those which may actually control the robot. (Note that several subsumption networks may control a robot, so that several different behavior modules may be generating actions at the same time for different subsystems). This type of hierarchical selection lends itself to the use of schema theory and object-oriented software techniques.

The arbitration network composed of suppression, inhibition, and default nodes in the subsumption architecture functions similarly to the schema assemblage construct in schema theory. A schema assemblage is a collection of (possibly interconnected) schemas and is itself regarded as yet another schema. The component schemas are definitions rather than instantiations, so that multiple inclusions in different schemas are realizable. This self-similar hierarchical structure lends itself to object-oriented software construction techniques. Object classes are built from less specific classes as specializations of those classes in the same way that subsuming behaviors are triggered by special cases of the triggering stimuli of subsumed behaviors.

The partial world model relevant to a subset of behavioral modules is contained in the module which manages that subset. This is the highest-level module contained in that subset. For example, a behavior called **go-to(elevator)** which goes to the elevator in a building given the current location must direct lower level behaviors to orchestrate a sequence of behaviors (i.e., make managerial decisions) which are triggered at intersections, such as **turn(left)**, **turn(right)**, and **turn(straight)**. The **go-to** behavior contains the necessary world knowledge to get to a known landmark, in this case the elevator. The managed behavior (**turn**) contains no world knowledge of that nature. Its world knowledge is only that which it needs to know to perform a turn successfully.

5. Maze Navigation

Navigation through a maze provides a clear and simple example of the generation and execution of plans. In this case, the successful path through the maze is directly representable as a path through state space, the relevant state in this case being location. A navigation plan is a predetermined sequence describing the choices made at each decision point, i.e., at forks in the road or intersections. This can be represented as a traversal of a decision tree (Fig. 2).

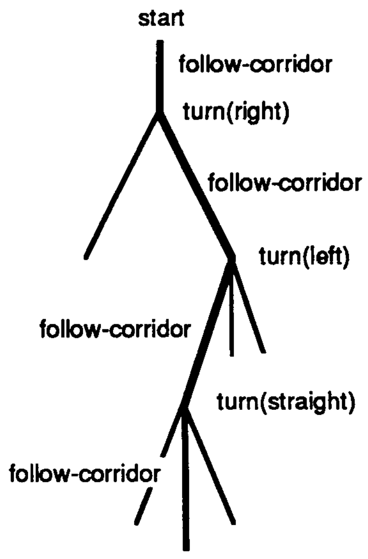

Fig. 2. Maze navigation decision tree

Upon encountering a decision point, such as an intersection, the subsumption architecture relies on a preprogrammed, rote strategy to select one of the alternatives. This arbitrary tie-breaker rule may be "always head south", or "follow the left wall". These can be successful strategies for maze navigation, but they are rarely optimum.

The best plans are based on previous experience. Searches for optimal paths through state space have been studied extensively in artificial intelligence. These require a cost criterion as well as a guide to choices available, i.e., a map. A map simply represents the accumulated experience of the mapmaker. Heuristics may be used (i.e., stay on the main path, or go straight, until you have a good reason not to) in the absence of specific map knowledge.

This search for an optimum path creates a plan based on the expectations of the planner. If the domain has changed, or the map is incorrect, the planner must generate a new plan. The new plan, in order to be optimal, may require some backtracking, or it may start from the current state[23]. The subsumption planner replans, without explicit backtracking, from the moment it senses a discrepancy between expectation and reality. The plan generated may include as its first segment a backtrack; however, from the point of view of the new plan, it is not backtracking, but rather generating a new complete plan from the current state.

The current location and orientation of the robot is represented as a point in (abstract) state space. Subspaces representing the sensors and their abstractions which are used to determine location, orientation, obstacles, and other properties important to navigation and travel, will contain the triggers for these behaviors. A behavior module is triggered when the current state lies within some specific trigger region. For example, the region of sensor state space which represents the situation in which a large object is directly in front of the robot should trigger an obstacle avoidance behavior module which causes a turn.

5.1 Path planning

A goal is the desired end state of a behavior. This definition of a goal assumes that the robot has achieved the purpose for its existence when this end state has been reached. This is only the case for simple robots and well-defined behaviors. In general, each subtask also has a goal, and each sub-subtask has goals. Similarly, the goal for the entire behavior is really a subgoal of a larger contextual plan, which may only be implicit in the design of a robot. For example, a robot's explicit goal at the topmost level may be to achieve a mowed lawn. This is a subgoal of the implicit task "keep the lawn mowed" which requires that the robot repeat the lawn mowing subtask whenever the grass gets too long. Real plans thus have a hierarchical, self-similar nature where simpler subtasks look very similar to the contextual task. Thus we can treat contextual plans in the same way as subtasks.

The decision tree which represents all possible paths (starting from the current state) to a goal is the problem space within which the navigation plan must be constructed. This problem space is a virtual scratchpad for the subsumption planner. The root of this tree represents the system's current location. The planner visualizes a future course of action which is expected to reach a goal.

The planner makes its decisions based on general knowledge of the problem domain, which it has accumulated by experience or by design. The benefit of using a planner over simple reactive behavior is that knowledge which is more global in nature than that available at the decision point may be applied to solve the problem in (hopefully) a more efficient manner. This more-global knowledge is in fact a partial world model. The premise of the subsumption planner is that partial world models may be distributed across the system as cause-effect predictors, associated with the action-generating behaviors which are capable of steering the robot into a desired state.

It is the responsibility of the cause-effect predictors to propagate the decision tree into the future to determine the best plan. Unlike a game theoretic min-max planner, in which the goal is to win the game, not to reach any specific game state, the subsumption planner can create a much more specific visualization of the goal. A chess playing algorithm can only "visualize" the winning goal state by forward chaining from the current state; there are a great number of possible winning states. It does not select a goal state and attempt to generate a plan which achieves it. The chess player tries a great number of state trajectories until a completely defined state is achieved which belongs to the subset of checkmate (goal) states. To the general purpose planner, however, most specifics of the goal state are irrelevant and can safely be ignored.

In the case of the maze navigator, the only aspect of the goal state which *is* relevant is the location. All other aspects of the goal state can be ignored. Since the goal state may be so loosely defined, the planner can restrict its search for potential behaviors to only those whose effect is to cause the robot to change location.

The most general behavior which can cause this effect may be called the **travel** behavior. This behavior may direct, manage, or simply allow (but not micro-manage!) the action of subservient behaviors which cause forward motion, turns, obstacle avoidance, etc.

The entire course of action of the **travel** behavior need not be planned in great detail in advance. The robot need only realize that **travel** is capable of producing the desired effect. Once this is known, the robot depends on **travel** to complete the entire task of getting to the goal. The repertoire of behaviors **travel** has at its beck and call will perform their own planning as needed.

The **travel** behavior makes its decisions in the context of its view of the sensor space. It delegates responsibility for selecting a sequence of turns at intersections to the **navigate-maze** behavior when it senses the domain (walls and passageways) through which it must travel. (An alternate behavior might be **navigate-meadow**; a meadow is a term for a large area where there is a possibility that proximity sensor bearings may be lost, so other navigation techniques, such as dead-reckoning, are called for.)

The **navigate-maze** behavior selects an active subordinate behavior based on its "map" (a representation of its expectations) or on heuristics if the map does not apply to the current situation. It does this by examining relevant elements of the real or virtual state space (orientation and location) and associating with that trigger state a "best" reaction. The "map" is therefore in the form of an associator which generates some reaction based on a set of inputs. This associator represents a partial world model. Each managerial behavior module contains an associator which embodies only the knowledge relevant to that behavior.

The plan, represented as a trajectory in state space, has associated with it a quality indicator. When the state arrives at a decision point in real execution, the decision it makes is the one associated with the highest quality indicator. If sensors indicate a deviation from the expected sensor state as contained in the virtual sensor space, the quality of trajectories which are affected by that deviation is reduced. Alternate plans may or may not be generated at that point. At the point of departure from expectations (i.e., the current state), the robot makes a decision according to the new highest quality plan.

5.2 An Example

Assume that a maze is to be navigated as in Fig. 3. The robot will start at location *s* and is to find its way to goal location *g*. There are two paths. The shortest path, right at *a* and left at *b*, is blocked by an obstacle which cannot be sensed until past *b*.

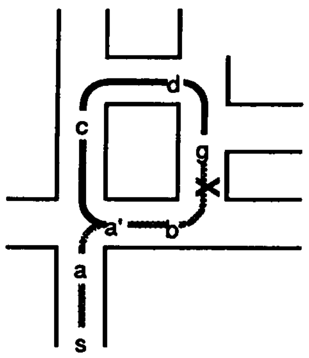

Fig. 3. Maze example

The subsumption planner contains a **travel** behavior capable of physically relocating the robot. This manager behavior controls subordinate behaviors **navigate-maze** and **navigate-meadow**. In the subsumption architecture, it would do this by inhibiting output from the undesired behavior. Another behavior, **follow-corridor**, is a behavior triggered upon sensing walls on either side and clear space in front. It simply travels along the midline of the corridor. The **turn** behavior is a composite behavior, consisting of a **turn-manager** and four types of subordinate turn generating behaviors, **turn(left)**, **turn(right)**, **turn(straight)**, and **turn(back)**. These turns are dependent on orientation; turns dependent on compass direction could be used identically. The **turn-manager** contains a turn-associator which is the mechanism containing the world model for the navigation domain. Since navigation in the maze world is performed by deciding which way to turn at intersections, the turn-map in this case resides in **turn-manager**. Another partial model, the corridor-length-map, provides expectations of distance to the **follow-corridor** predictor. Other low level modules, not discussed here, prevent collisions with obstacles, keep the robot in the center of the hall, keep the robot moving, etc.

The robot will behave as follows:

1) The robot begins at location *s*. The **follow-corridor** reactor begins to generate actions as in the traditional subsumption architecture, and the robot starts moving down the hall. It moves toward location *a*.

2) At the same time that the reactor begins generating actions, the **travel** behavior is triggered by some higher level behavior to start the planning process. The **travel** behavior assumes responsibility for moving the robot from *s* to *g*. The predictor generates a simple state space trajectory between *s* and *g*. **Travel** allows **navigate-maze** to assume re-

sponsibility for completing the plan and its execution. Only **navigate-maze** is triggered by the abstract maze/meadow sensor. The **follow-corridor** sensor is also triggered by the abstract corridor sensor. The **follow-corridor** predictor generates the expected location *a* based on its knowledge of corridor length. In the virtual sensor map, *a* is stored as a waypoint to be reached at time t_q The virtual sensor space at *a* indicates the sensory situation of a 4-way intersection. The **turn-manager** behavior is virtually triggered by an abstract intersection sensor.

3) The **turn-manager** associator senses global location *a*. The global location sensor is abstracted from a variety of fused raw sensors, perhaps including visual clues, wheel odometers, spatial sensors such as sonar or rangefinder, and any other sensors which distinguish this location as different from others in any way. The associator outputs "turn-right" as the highest quality plan, followed by "turn-straight" as the next highest. Note these are not behaviors but rather decisions on which behavior should be activated as part of the plan. Other alternatives have a quality indication of zero.

4) The planner is now done until the robot reaches *a*, since all details for the plan segment from *s* to *a* are complete. The turn manager inhibits its own triggering until ready to resume planning.

5) The robot reaches *a*. Since the prediction turned out to be accurate, the top quality plan has not changed. This plan indicates the behavior **turn(right)** should be executed next. Turn right is selected by **turn-manager** as active, and the predictor generates *a'* as the next waypoint. *a'* virtually triggers **follow-corridor**, and that predictor generates waypoint *b*. Virtual sensor situation *b* then triggers **turn-manager**, which inhibits itself from making a commitment until *b* is physically reached. The planner waits for the robot to catch up.

6) At *b*, the **turn-manager** associator generates the decision "turn-left". Robot starts the **turn(left)** behavior using the reactor mechanism. But, an unexpected obstacle is sensed. The blocked path causes the entire trajectory quality to go to zero. The expectation violation alert is made available to any behavior.

7) The robot is still near *b*. The **navigate-maze** behavior notices the expectation violation. Its reaction is to replan. It predicts that it can still reach the goal, since there is a second, albeit lower quality, plan available from a previous waypoint. All motion-generating behaviors are allowed to trigger from the real sensor state *b*. **Follow-corridor** has the best quality plan, a direct trajectory to end up near *a*. The robot now moves under direction of **follow-corridor**.

8) At *a'*, *c*, and *d*, decisions are made in the same manner as above. Finally, the robot moves to *g*. **Travel** has completed its visualized plan, along with all its subordinates. The **travel** behavior terminates, and current state *g* may be used to trigger the next behavior.

6. Conclusion

This paper has described an extension to the subsumption architecture which implements planning using a virtual sensor space as a planning tool. The planner, given a goal, generates a new plan, monitors execution of the plan, and replans in the event of a discrepancy between expectation and reality. The planner operates as a distributed network in parallel with the existing subsumption network. The theoretical motivation for this technique was presented. An example in the domain of maze navigation showed its operation in a simple application.

References

[1] James S. Albus, *Brains, Behavior, and Robotics*, BYTE Books, Peterborough, NH, 1981.

[2] Marvin Minsky, *The Society of Mind*, Simon and Schuster, Inc., New York, 1986.

[3] Michael A. Arbib, *Schema Theory*, University of Southern California Technical Report 91-03, 1991.

[4] Eugene C. Chalfant and Les Gasser, *Describing Computation and Knowledge using Attractor Superdynamics: Preliminary Report*, University of Southern California Distributed Artificial Intelligence Group Research Note 62, 1990.

[5] Rodney Brooks, *A Robust Layered Control System for a Mobile Robot*, IEEE Journal of Robotics and Automation, Vol. RA-2, No. 1, March 1986.

[6] Rodney Brooks, *Intelligence without representation*, Artificial Intelligence, Vol. 47, pp. 139-159, 1991.

[7] Rodney Brooks, Challenges for Complete Creature Architectures, in *From animals to*

animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior, Jean-Arcady Meyer and Stewart W. Wilson, eds., The MIT Press, Cambridge MA, 1991.

[8] Jonathan H. Connell, *A Behavior-Based Arm Controller*, IEEE Transactions on Robotics and Automation, Vol. 5, No. 6, December 1989.

[9] Jonathan H. Connell, *Minimalist Mobile Robotics: A Colony-style Architecture for an Artificial Creature*, Academic Press, Inc., Boston, 1990.

[10] Maja J. Mataric, *Integration of Representation Into Goal-Driven Behavior-Based Robots*, IEEE Transactions on Robotics and Automation, Vol. 8, No. 3, June 1992.

[11] Marco Dorigo and Uwe Schnepf, *Genetics-Based Machine Learning and Behavior-Based Robotics: A New Synthesis*, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 23, No. 1, January/February 1993.

[12] N. Tinbergen, *The Study of Instincts*, Oxford University Press, Oxford, England, 1966.

[13] D. M. Lyons and A. J. Hendriks, *Planning by Adaptation: Experimental Results*, IEEE International Conference on Robotics and Automation, submitted for review.

[14] Patrick J. Hayes, *The Frame Problem and Related Problems in Artificial Intelligence*, in "Readings in Artificial Intelligence", Bonnie Lynn Webber, Nils J. Nilsson, eds., Tioga Publishing Company, Palo Alto, 1981.

[15] Elaine Rich, *Artificial Intelligence*, McGraw-Hill, New York, 1983.

[16] Edmund H. Durfee and Victor R. Lesser, *Using Partial Global Plans to Coordinate Distributed Problem Solvers*, Proceedings of the 1987 International Joint Conferences on Artificial Intelligence, pp. 875-883, 1987.

[17] Edmund H. Durfee, Victor R. Lesser, and Daniel D. Corkill, *Coherent Cooperation Among Communicating Problem Solvers*, IEEE Transactions on Computers, C-36, pp. 1275-1291, 1987.

[18] Edmund H. Durfee and Victor R. Lesser, *Partial Global Planning: A Coordination Framework for Distributed Hypothesis Formation*, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 21, No. 5, September/October 1991.

[19] Mitsuo Kawato, Yoshiharu Maeda, Yoji Uno, and Ryoji Suzuki, *Trajectory Formation of Arm Movement by Cascade Neural Network Model Based on Minimum Torque-change Criterion*, Biological Cybernetics, pre-press correspondence, 1990.

[20] Masazumi Katayama and Mitsuo Kawato, *Virtual Trajectory and Stiffness Ellipse During Force-Trajectory Control Using a Parallel-Hierarchical Neural Network Model*, Proceedings of the Fifth International Conference on Advanced Robotics, Italy, 1991.

[21] Makoto Hirayama, Mitsuo Kawato, and Michael I. Jordan, *Speed-Accuracy Trade-off of Arm Movement Predicted by the Cascade Network Model*, ATR Auditory and Visual Perception Research Laboratories Research Note NC89-65, Kyoto, Japan, 1990.

[22] Sukhan Lee, Paul S. Schenker, and Jun Park, *Sensor-Knowledge-Command Fusion Paradigm for Man/Machine Systems*, Proceedings of SPIE International Symposium on Advanced Intelligent Systems, Boston MA, 1990.

[23] Bruce Abramson, *A Decision-Theoretic Framework for Integrating Sensors into AI Plans*, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 23, No. 2, March/April 1993.