AuRA

(Autonomous
Robot
Architecture)

Contents

About the paper

This paper is a final project on course "Architectures of Autonomous Systems" developed and told by Dr. Reuven Granot at faculty of Mathematics and Computer science at Haifa University.

The paper is prepared by Goihman Ina, a third year student.

In this paper the structure of AuRa architecture is described.
Chapter 4 is not necessary, but I find it to be interesting. So I decided to include it and to deepen in the theme.

Chapter 1

About Hybrid Architecture

There are many ways to structure a robot, yet every one will fall into one of the basic architecture control systems we've studied at the course:

1. Reactive control - no look ahead, react (time-scale).
2. Deliberative control - look ahead, think and plan, then act.
3. Behavior-based control - distribute thinking over acting.
4. **Hybrid control - combine reactive with deliberative, think slowly, react quickly.**

Hybrid control ***is a combination of reactive and deliberative control. Hybrid robotic architecture states that a union of deliberative and reactive approaches can potentially yield the best of both worlds.***

A modern hybrid system typically consists of three components: a reactive layer, a planner and a layer that outs the two together.
A hybrid system must combine the two time-scales and representations of the deliberative and reactive architectures in an effective way. Usually, a middle layer is required. Its mission is to achieve the right compromise between the two ends. Consequently hybrid systems are often called the 'Three-layer-architecture'.

The three-layer architecture is not the only possible combination of a reactive module with a deliberative planner. It belongs to a hierarchical integration of planning and reacting with different activities and different time scales. Planning and reaction may be coupled as concurrent activities as well as the planner may guide reaction by configuring and setting the parameters of the reactive control module.

There have been several methodologies used to integrate reactive and deliberative layers in hybrid architectures for autonomous agents. Each has its own particular strengths and weaknesses, and none can be said to be the best. In fact, it has been noted (Nwana 1996) that hybrid architectures generally suffer from at least three high-level problems:

● unprincipled designs that not only require hand crafting (creating) of each control module, but may also overlap functionality

● application-specific tasks, which may very well be unavoidable

● unspecified supporting theory

There are also practical difficulties with representations between layers, error-detection across layers, and timing issues related to the different time scales on which the layers operate.

Several Hybrid Approaches Have Been Developed:

- **AuRA** (Arkin 1986)
- Atlantis (Gat 1991)
- Sensor-Fusion Effects (SFX) (Murphy 1996)
- 3-Tiered (3T) (JPL1990s)
- Saphira (Konolige 1998)
- Tack Control Architecture (Simmons 1997)
- Planner-Reactor (Lyons and Hendriks 1992)
- Procedural Reasoning System (PRS) (Georgeff and Lansky 1987)
- SSS (Connell 1992)
- Multi-Valued Logic (Saffiotti 1995)
- SOMASS Hybrid Assembly System (Malcom and Smithers 1990)
- Agent Architecture (Hayes-Roth 1993)
- Etc., Etc.

Representative Hybrid Architectures:

♦ **Selection**:	Planning is viewed as configuration - **AuRA**
♦ **Advising:**	Planning is viewed as advice giving – **Atlantis**
♦ **Adaptation:**	Planning is viewed as adaptation of controller - **Planner – Reactor**
♦ **Postponing**:	Planning is viewed as a least commitment process – **PRS**

<u>AuRA</u>

Planning and execution components:
a **hierarchical system** → mission planner, spatial reasoner, and plan sequencer.

- Traditional: highest level is a mission planner → establishes **high - level (global) goals + constraints**.
- Mission planner → acts as an interface to human operator.
- Spatial reasoner → navigator uses knowledge stored in Long Time Memory to construct **a sequence of path legs** that the robot must execute.
- Plan sequencer (pilot) →translates each path leg into a set of motor behaviors for execution.
  - In the original implementation it was a rule based system.
  - Now a finite state sequencer was implemented.

coupled with a **reactive system** → the schema controller.

The collection of behaviors (schemas), specified by the sequencer, is sent to the reactive system for execution.

At this point, **deliberation ends/stops**, and **reactive execution begins.**

AuRA is a hybrid architecture that attempts to combine a schema-based reactive layer with non-schema-based deliberative extensions.

Atlantis

Control Layer:

Reactive controller charged with managing collection of primitive activities.

Implemented in ALFA, a LISP based program language, similar to Rex, a circuit based language.

Sequencing Layer:

Modeled after **RAP** (Reactive Action Packages, Firby 1989).

RAP is a situation-driven execution reactive method, in which the current situation **provides an index** into a **set of actions** regarding how to act in that environment.

Conditional sequencing occurs upon the completion of subtasks or detection of failure.

Notion of *cognizant failure* was introduced, referring to the **robot s ability to recognize on its own when it has not or cannot complete its task**: it has knowledge about its failures.

● Monitor (**task specific**) routines are added to determine if things are going as they should and then interrupt the system if cognizant failure occurs.

Deliberative Layer.

Deliberation occurs **at the sequencing layer request**.

Consists of traditional LISP based AI planning algorithms, **specific to the task** at hand.

The planner s output is viewed only as advice to the sequencer layer: it is not necessarily followed or implemented.

Design proceeds from bottom up:

low-level activities capable of being executed within the reactive controller are first constructed.

suitable **sequences** of primitive behaviors are then constructed.

followed by **deliberative methods that assist** in the decision

done **by the sequencer**.

Planner-Reactor

A **suboptimal reactor** may be present at **any time.**

The **planner s goal is to improve the performance** of the reactor at all times (any-time planning).

Any-time planners provide approximate answers in a time critical manner:

* at any point a plan is available for execution, and

* the quality of the available plan increases over time.

Situations provide the framework for structuring sets of reactions.

* can be defined hierarchically, as behavioral structures for use **in the reactor** and **not specific** robotic commands.
* denote the state of the robotic agent is currently in (regarding a task).

PRS

Plans are the primary mode of expressing action.

They are **continuously** determined in reaction to the current situation.

Previously formulated **plans undergoing execution** can be interrupted and abandoned at any time.

Representations of the robot s **beliefs, desires, and intentions** are all used to formulate a plan.

The plan represents the **robot s desired behaviors** instead the traditional AI planner s output of goal states to be achieved.

The **interpreter** drives system execution, handling the plan switching.

A symbolic plan always drives the system.

it is **not reactive in the normal sense** of tight sensori-motor pair execution

it **is reactive in the sense** that perceived changing environmental conditions permit the robotic agent to alter its plans **on the fly.**

Chapter 2

In short-What is AuRA?

Autonomous Robot Architecture (AuRA)

AuRa is an Autonomous Robot Architecture that was developed at College of Computing Mobile Robot Laboratory at Georgia Tech. In AuRA the overarching hybrid deliberative/reactive architecture is employed, motor schemas provide the reactive component of navigation.

Instead of planning by predetermining an exact route through the world and then trying to coerce the robot to follow it, motor schemas (behaviors) enable the robot to interact successfully with unexpected events while still striving to satisfy its higher level goals.

Motor schema outputs are analogous to potential fields. Multiple active schemas are usually present, each producing a velocity vector driving the robot in response to its perceptual stimulus. The robot only needs to compute the single vector at its current location. Each of the individual schemas posts its contribution to the robot's motion at a centralized location. The resultant vectors are summed and normalized to fit within the limits of the robot vehicle, yielding a single combined velocity for the robot. These vectors are continually updated as new perceptual information arrives, with the result being immediate response to any new sensory data.

The advantages of this form of navigation are many.
They include rapid computation and the ability to be mapped onto parallel architectures, making real-time response easily attainable. Modular construction affords ease of integration of new motor behaviors simplifying both system maintenance and the ease of transfer to new problem domains. Motor schemas readily reflect uncertainty in perception, when such a measure is available, and also react immediately to environmental sensor data. These factors all contribute to the needs of a navigational system that will successfully assist a robot's intentional goals.

Chapter 3

AuRA - detailed explanation

3.1 Introduction

The Autonomous Robot Architecture (AuRA) was developed in the mid -1980's as a hybrid approach to robotic navigation.
Arkin was among the first to advocate the use of hybrid, deliberative (hierarchical) and reactive (schema-based), control systems within the Autonomous Robot Architecture.

Arkin found that specific robot configurations can be constructed that integrate behavioral, perceptual, and a priori environmental knowledge. This can be done while incorporating a planner that could reason over a flexible and modular behavior-based control system.
Hybridization in this system arises from the presence of two distinct components: a deliberative hierarchical planner, based on traditional artificial intelligence (AI) techniques; and a reactive controller, based upon schema theory.
Arkin's was the first robot navigational system to be presented in this integrative manner.

3.2 Overall Structure

Two components of planning and execution are present:
A **hierarchical deliberative system**, consisting of a **mission planner**, **spatial reasoner**, and **plan sequencer**, coupled to a **reactive system**, the **schema controller**.

In simple words:

The deliberative layer (hierarchical system) consists of:
Mission planner, that plans what to do;
Spatial reasoner (cartographer), that accesses the map to decide where to move;
Plan sequencer, that translates the decision of spatial reasoner to the schema controller for execution.
The reactive layer consists of:
Perceptual schemas, which receive inputs from the sensors, processes them and sends it to schema controller
Schema controller, that processes all the inputs, and commands to move the legs, arms, wheels and etc.

3.2.1 Mission Planner

The **Mission Planner**, like in traditional hierarchical planning system, is the highest level of AuRA, that establish high level goals for the robot and the constraints within which it must operate.

In AuRa-based system constructed to date, the Mission Planner has acted primarily as an interface to human commander. This is done by using assemblage. We'll see later more detailed explanation in Chapter 4.3.

3.2.2 Spatial Reasoner

The **Spatial Reasoner** use cartographic knowledge stored in long-term memory to construct a sequence of path legs, which the robot must execute to complete its mission.

In the first implementation of AuRA, this planner used the **A* algorithm** to search over a meadow map representation (hybrid free space/vertex graph).

A* Search
Best-first search, using f **as the evaluation function** and an **admissible *h* function**.
Evaluation function *f (n)* = estimated cost of the cheapest solution through *node n.*
$f(n)= g(n)+ h(n)$
$g(n)$ gives the path cost from start to node n
$h(n)$ is the estimated cost of the cheapest path from n to the goal.

Example: estimate the distance between to cities by the straight-line distance.

A*	
Exhibit monotonicity	(along any path from the root *f* never decreases).
Optimally efficient	(algorithm that extend search paths from the root).

Note:
A* Search Algorithm does not guaranty that the way will be the shortest; moreover it does not guaranty that the way will be found, however the probability is high (it is possible, that the ways that lie on straight line between the cities, do not connect ,in the end, one city to another .

Router Spatial Reasoner
In 1990 the original A* Spatial Reasoner has been replaced with **Router**, a multi-strategy adaptive navigation path planner.
Mission planner generates a specific mission plan and identifies specific path planning tasks. Than passes it to Router. Given a specific path planning task, Router uses a combination of **model based** and **case based** methods to solve it.
In the **model-based mode**, the system plans paths by heuristically searching a hierarchical model of the navigation space; and in **the case-based mode**, it plans new paths by adapting and combining previously planned paths.

3.2.3 Plan Sequencer

The **Plan Sequencer** translates each path leg generated by the Spatial Reasoner into a set of motor behaviors for execution.
In the original implementation, the Plan sequencer was a **rule-based system**.
It means that Plan Sequencer has a ready "formula" for every possible path leg. Rule-based systems do not perform any actual computations on the data. The rules consist of linguistic variables and look like (C = T1 or T2)…

More recently it has been implemented as a **finite state sequencer (acceptor)**.
A **Finite State Acceptor** is used for arbitration on states. A state is an activation of certain schemas.

The **Finite State Machine** consists of a set of states (including the initial state), a set of input events, a set of output events, and a state transition function. The function takes the current state and an input event and returns the new set of output events and the next state.

The state machine can also be viewed as a function that translates an ordered set of input events into a corresponding set of output events.

Finite state acceptors (FSA) diagrams are used to represent a sequence of complex behaviors, where each state represents an assemblage of schemas (see Chapter 4.2).
This type of control differs from the reactive system that does not require arbitration between behaviors. The control provides a convenient mechanism for the achievement of multiple concurrent goals.
It has no layering of behaviors and consists of dynamic collection of schemas instantiated for the current context. As the context changes, the planning component can alter the behavioral composition. Learning and adaptation are also easily introduced through the flexibility afforded at this level.

Finite State Machine can be deterministic, nondeterministic and probabilistic.
A **deterministic** FSM (**DFA**) is one where the next state is uniquely determined by a single input event.
A **nondeterministic** FSM (**NFA**) is one where the next state depends not only on the current input event, but also on an arbitrary number of subsequent input events. Until these subsequent events occur it is not possible to determine which state the machine is in.
In a **probabilistic** FSM, there is a predetermined probability of each next state given the current state and input.

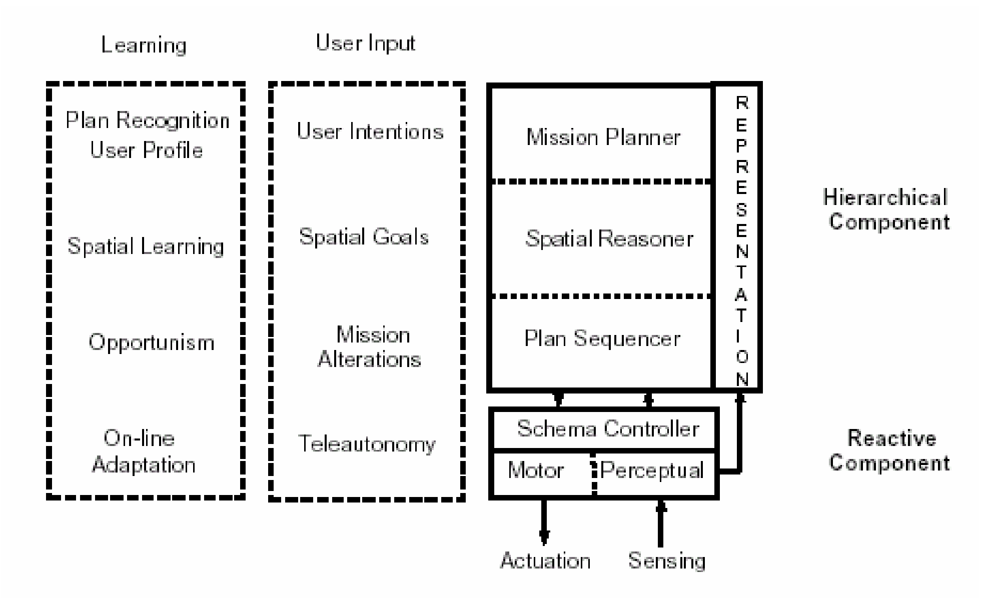
Finally, after translating a path leg into a set of motor behaviors, the Plan Sequencer sends the collection of behaviors (schemas) to the robot execution.
At this point, deliberation stops, and reactive execution begins.

How all this work in simple words?
For example, the Mission Planner commands to the Spatial Reasoner to reach green wall in the building, something like “go to the end of the corridor, turn left and move up the stairs, look for green color”.
Spatial Reasoner is looking for the map of the building and other useable information in Long Term Memory. Then it composes a set of paths and sends it to the Plan Sequencer.

Plan Sequencer receives input to move in accordance with this special path. It translates each path leg generated by the Spatial Reasoner into a set of motor behaviors for execution. It looks like "go forward, turn left, avoid moving objects, move up… The FSA decides by arbitration what the final set of the behaviors is. After this the Plan Sequencer sends the set to motor schema manager in the reactive layer.

Then motor schemas generate vectors. For example, "move up" generate vector with some velocity and direction, "turn left" has his own vector and so on. Schemas send asynchronously the vectors to the schema manager. Sometime conflict situations appear. The solution is found by vector addition. And then the resultant vector is sent for execution, for example turn left 45 degrees.

Figure 1: High-level AuRA Schematic



3.2.4 Schema manager

The **motor schema manager** (controller) is responsible for controlling and monitoring the behavioral process at run-time. It processes the input from **motor schemas** and Plan Sequencer.

As we said, the Plan Sequencer (pilot) is charged with implementation of path legs.
To do so, the pilot chooses from a repertoire of available sensing strategies and motor behaviors (schemas) and passes them to the **motor schema manager** for instantiation.

Distributed control and low-level planning occur within the confines of the motor schema manager during its attempt to satisfy the navigational requirements.

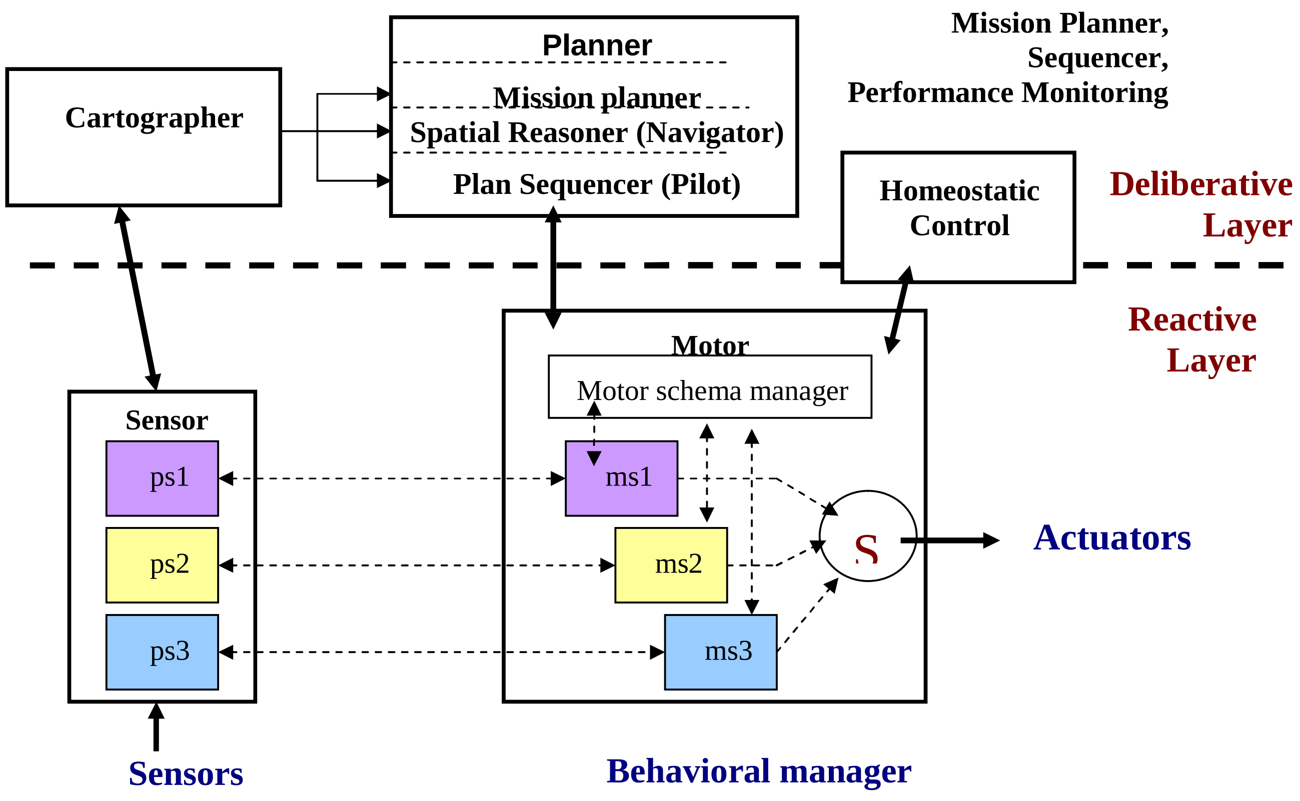
As the robot proceeds, the cartographer, using sensor data, builds up a model of the perceived world in short-term memory. If the actual path deviates too greatly from the path initially specified by the navigator (Spatial Reasoner), the navigator will be reinvoked and a new global path computed.
If the deviations are within acceptable limits, (as determined by higher levels in the planning hierarchy), the pilot and **motor schema manager** will, in a coordinated effort, attempt to bypass the obstacle, follow the path, or cope with other problems as they arise.

So how all this is done in more detailed way?

Motor schemas receive input from perceptual schemas, which receives input from the sensors.
After processing the input information, each of the schemas generates a <u>response vector</u> in a manner, analogous to the <u>potential field method</u>.
Then every schema transmits its result (a vector) to a process (move-robot).
This process is a part of **motor schema manager**.
Because schemas can operate asynchronously, the process sums and normalizes the inputs (vectors) and transmits them to the low-level control system for execution.
This is the basis for reactive navigation.

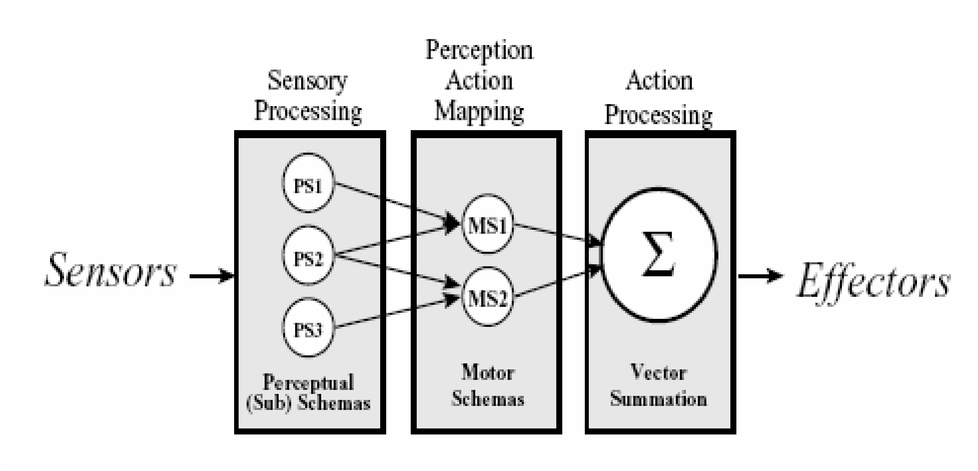
Let us see at the pictures how Aura is constructed:

Figure2: **AuRA structure**



As we mentioned above, the **reactive layer uses** two classes of schemas**, a perceptual and a motor schemas**. Its structure is shown in

Figure 3: **Basic schema-based reactive architecture**



Within AuRA a **homeostatic control system** (tested only in simulation to date) is interwoven with the **motor** and **perceptual schemas**. The idea for this system was taken from the animals' endocrine system..

Internal sensors, such as fuel level and temperature transducers, provide internal messages that change the overall motor response's performance. They alter the relative strengths of behaviors and internal parameters in order to maintain balance and system equilibrium (homeostasis).
We'll speak more about it in chapter 4.2, because it's a rather interesting theme.

Once reactive execution begins, the deliberative component is not reactivated unless a failure is detected in the reactive execution of the mission.

<u>In case of failure</u>

What is a failure in AuRa?
It is a lack of progress, caused by a velocity of zero or a time-out.
In case of failure in the reactive execution of the mission the deliberative component is **not** reactivated.
At this point hierarchical planner is reinvoked one stage at a time from the bottom up, until the problem is resolved.
First, the **Plan Sequencer** tries to reroute the robot. It uses the information that has been obtained during navigation and stored in short-term memory (STM). Original implementations used sonar maps produced using the Elves-Moravec algorithms for spatial world modeling.
If for some reason this is unsatisfactorily, the **Spatial Reasoner** is reinvoked. It tries to regenerate a new global route that bypasses the affected region entirely.
If this still is unsatisfactory, the **Mission Planner** is reinvoked. It informs the operator of the difficulty and asks for reformulation or abandonment of entire mission.

3.3 Motor schemas and vectors

Motor schemas are a reactive component of AuRa, developed by Ronald Arkin (Ga Tech).
A schema is a behavior and can be presented as a function, written in some programming language.

Motor Schemas are based upon schema theory:

Developed by **Arbib**, formalized by **Lyons**, and operationalized by **Arkin**, schema theory develops sensori-motor coordination as a series of active concurrent processes, each independently striving to achieve an agent's goals.

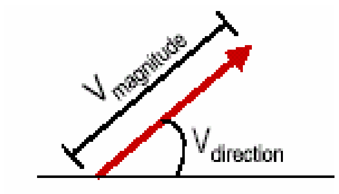
Motor Schema based upon schema theory:

--Explains motor behavior in terms of concurrent control of many different activities
--Schema stores both how to react and the way that reaction can be realized
--A distributed model of computation
--Provides a language for connecting action and perception
--Activation levels are associated with schemas that determine their readiness or applicability for acting
--Provides a theory of learning through acquisition and tuning

Output vector consist of both orientation and magnitude components:

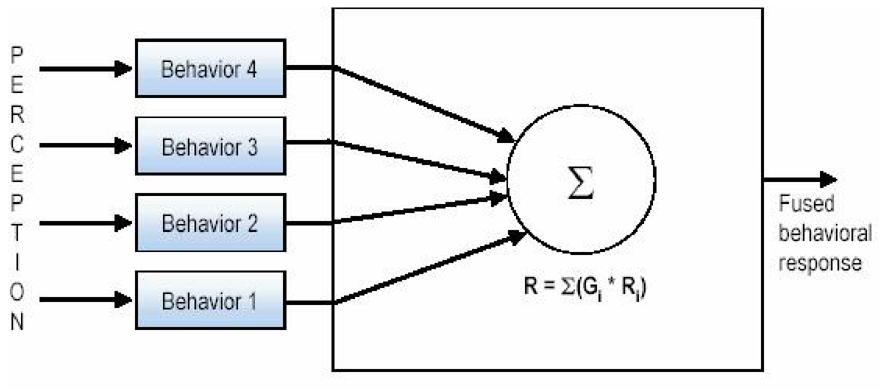
$V_{magnitude}$ denotes magnitude of resultant response vector

$V_{direction}$ denotes orientation



But what we do if we have many behaviors and each of them tells to do something else? Every behavior is producing a different vector! (conflict resolution)
All the recommendations (vectors) are combined together (summed) into a "**potential field**" to produce the actual movement of the robot.

Motors schemas achieve behavioral fusion via vector summation:



Behavioral fusion

Many primitive robotic behaviors have been implemented within AuRA. (A primitive robotic behavior is a computable function, implemented in some convenient programming language).
Some of the motor schemas developed to date include:
● **Move-ahead:** move in particular compass direction.

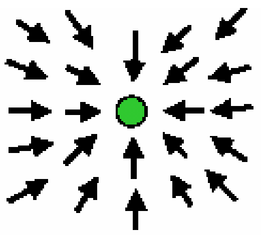
● **Move-to-goal** (both ballistic and guarded)**:** move towards a discrete stimulus

- **Stay-on-path:** move towards the center of a discernible pathway, e.g., a hall or road.
- **Avoid-static-obstacle:** move away from non-threatening obstacles.
- **Dodge:** sidestep approaching ballistic objects.
- **Escape:** Evade intelligent predators.
- **Noise:** move in random direction for a fixed amount of time (persistence)
- **Avoid-past:** move away from recently visited areas.
- **Probe:** move towards an open area.
- **Dock:** move in a spiral trajectory towards a particular surface.
- **Teleautonomy** - Introduce a human operator at the same level as other behaviors.

Each of this can be defined as a potential field of output vector responses.
Figure 4 shows some of this behaviors represented as fields.

Figure 4:

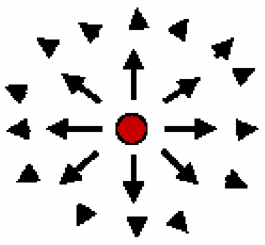
- Move-to-goal (ballistic):
  - $V_{magnitude}$ = fixed gain value
  - $V_{direction}$ = towards perceived goal



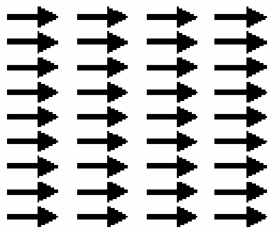
- Avoid-static-obstacle:

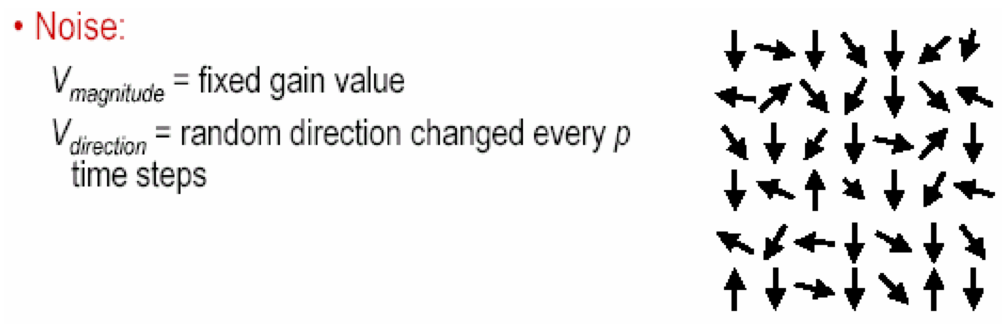
$$V_{magnitude} = \begin{cases} 0 & \text{for } d > S \\ \dfrac{S-d}{S-R} * G & \text{for } R < d \le S \\ \infty & \text{for } d \le R \end{cases}$$

where S = sphere of influence of obstacle
R = radius of obstacle
G = gain
d = distance of robot to center of obstacle



- Move-ahead:
  - $V_{magnitude}$ = fixed gain value
  - $V_{direction}$ = specified compass direction





• Noise:

$V_{magnitude}$ = fixed gain value

$V_{direction}$ = random direction changed every p time steps

The arrow at each point illustrates the vector the behavior would generate if the robot were at that point. **The robot does not actually compute the entire field; it only computes a vector based on its current local perception.**

Potential fields are known to suffer from local minima (vectors may sum to 0) and cyclic problems.
Solutions for Dealing with Local Minima

•Inject noise, randomness: – "Bumps" robot out of minima
•Include "avoid-past" behavior: – Remembers where robot has been and attracts the robot to other places
•Use "Navigation Templates": –The "avoid" behavior receives as input the vector summed from other behaviors
–Gives "avoid" behavior a preferred direction
•Insert tangential fields around obstacles

Potential fields in Motor schemas are a large and complicated theme. I think it's not good idea to explain it here, in this paper. I found well organized and explained material at:
http://www.cs.utk.edu/~parker/Courses/CS594-fall02/Lectures/Sept17.pdf

Differences of Motor Schemas versus Other Behavioral Approaches

•Behavioral **responses** are all represented **as vectors** generated **using** a **potential fields** approach
•**Coordination is** achieved by **vector addition**
•**No** predefined **hierarchy** exists for coordination; instead, behaviors are configured at run-time
•Pure **arbitration is not used**; each behavior can contribute in varying degrees to robot's overall response

As a conclusion:
Each desired behavior is represented by a Motor Schema.
Each motor schema is a separate, asynchronous computing "agent".
Each motor schema reacts to sensory information from the environment.
Each motor schema produces as output a vector that represents the suggested direction and speed of movement.

Schemas have many benefits, including:
● Concurrent actions (in a cooperative yet competing manner)
● Inexpensive computations (as only the current field need be calculated)
● Behavioral primitives (from which more complex behaviors can be constructed)
● Coarse grain granularity (for expressing the relationships between motor control and perception)
● Biological plausibility

Chapter 4

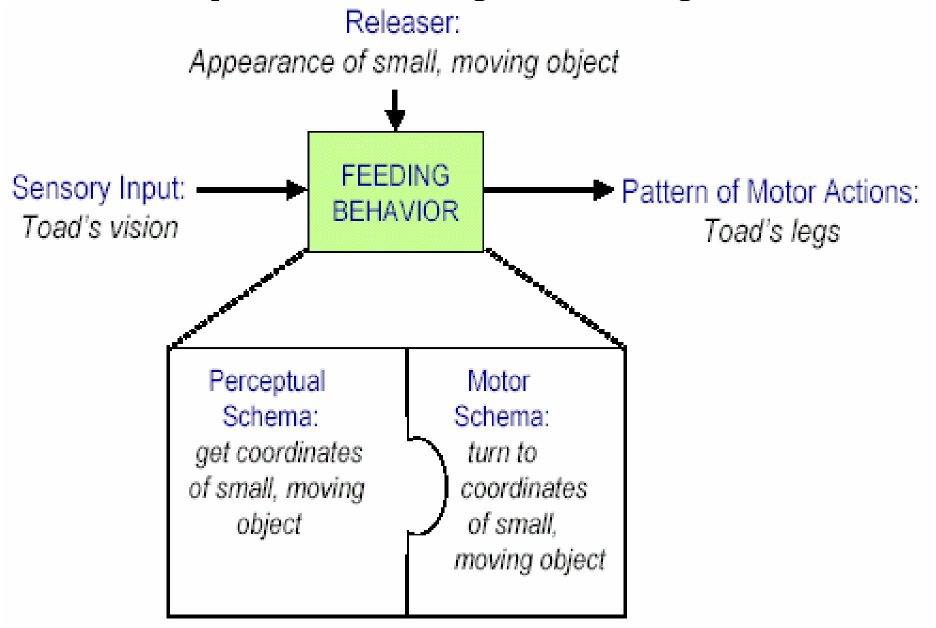
Every robot must act in a changeable environment. This is impossible to do without sensors.
We can divide the sensors into two groups, internal and external.
We've studied at the course and understand how external sensors act. IR or sonar sensors are comparatively simple, vision is much more complicated.
But we didn't talk about internal sensors.
Internal sensors are based on biological analogs in the animals' body.
Motor schemas have biological connections too.

4.1 Biological Connections, Homeostatic Control System

AuRa has been influenced by a wide range of ethnological, neuro-scientific, and psychological studies. The justification for **hybridization** of reactive and deliberative control can be found in studies by psychologists such as Norman and Shallice and Neisser. Action-oriented perception also has a strong psychological basis, motivated largely by Gibson's theory of affordances.

The most profound influence has been that of schema theory.
Within AuRa, schemas are employed at the reactive control level. They are encoded using the analog of the potential fields' method. Each motor schema receives sensory data from an associated perceptual schema and generates its reaction to the stimulus in the form of vector.
Biological similarities to this form of motor control have been observed in the navigation of **toad**, and limb control within the spinal cord of a **frog**.

Example of Toad's Feeding Behavior Using Schema


Releaser:
Appearance of small, moving object
Sensory Input:
Toad's vision
FEEDING BEHAVIOR
Pattern of Motor Actions:
Toad's legs
Perceptual Schema:
get coordinates of small, moving object
Motor Schema:
turn to coordinates of small, moving object

Homeostatic control system

The endocrine system in mammals, using hormones as chemicals messengers, serves as a mean of both information and processing control. The **homeostatic control system** was developed using models of the **mammalian endocrine system** as inspiration.

Negative feedback mechanism is fundamental to the control of endocrine system. It is very important that negative feedback control is an operation. Because, the system, when stressed, can be restored to a steady state, by the application of this control regime. This is crucial for the process of homeostasis.

1. Energy
Just as glucose fuels most of the cells of the body, some form of energy must be made available for the robot. The process of feeding the cells is done by glucose metabolism (changing insulin and hormone level that mark effects on all biological tissues). This aspect of energy management is highly suitable for incorporation into homeostatic control subsystem.
2. Temperature management
Warm-blood animals must maintain constant temperatures in their lifetimes always. Robots can act in suitable ranges of temperature. Of cause, robots mechanism of heating-cooling is not analogues to mammals (robots don't sweat).But the problems are the same:

a) **Global stress -** Heat must be exchanged between the robot and its surrounding to restore acceptable condition (e.g., robot is on the sunny part of Mercury).
b) **Local stress -** Heat must be redistributed within the robot to maintain reliable operation of a particular subsystem (e.g., overheating of robot arms while servicing a furnace).

In either case, temperature must be regulated by a control system as it occurs in mammals. Homeostatic control is used to manage this function.
3. Emergency notification
Messages are of two types.
The most common has its transmitter schema directly tied to the perceptual subsystem and its sensory processors. This is analogues to adrenalin that allows rapid increase in the rate of process.
Other emergency messages can arise from the motor controllers signaling when something is broken or otherwise gone awry.

Of cause, it is a lot to say about the signal schemas (receptor and transmitter schemas), about homeostatic control and biological connection of motor schemas. But this is another large theme.

4.2 Assemblage, MissionLab

Assemblage
Assemblages of behaviors are formatted by combining primitive behaviors in a meaningful ways. The relative importance of each behavior is encoded through the use of scalar multiplies applied to the output of each behavior. The resulting vectors are then combined using vector addition to yield the overall reaction of the robot to its environment.

From "Multi Agent Robots" by Cai Peng:

“Specifying a reactive behavioral configuration requires both a careful choice of the behavior set and the creation of a temporal chain of behaviors which executes the mission. This difficult task is simplified by applying an object-oriented approach to the design of the mission using a construction called an **assemblage** and a methodology called **temporal sequencing**.

The assemblage construction allows building high level primitives which provide abstractions for the designer. **Assemblages consist of groups of basic behaviors and coordination mechanisms that allow the group to be treated as a new coherent behavior.**
The assemblage is parameterized based on the specific mission requirements.
Assemblages can be reparameterized and used in other states within a mission or archived as high level primitives for use in subsequent projects. “

In simple words, the assemblage shows the behaviors and sets of behaviors in more comfortable way for the designer. This makes the human-robot interconnection possible and convenient.

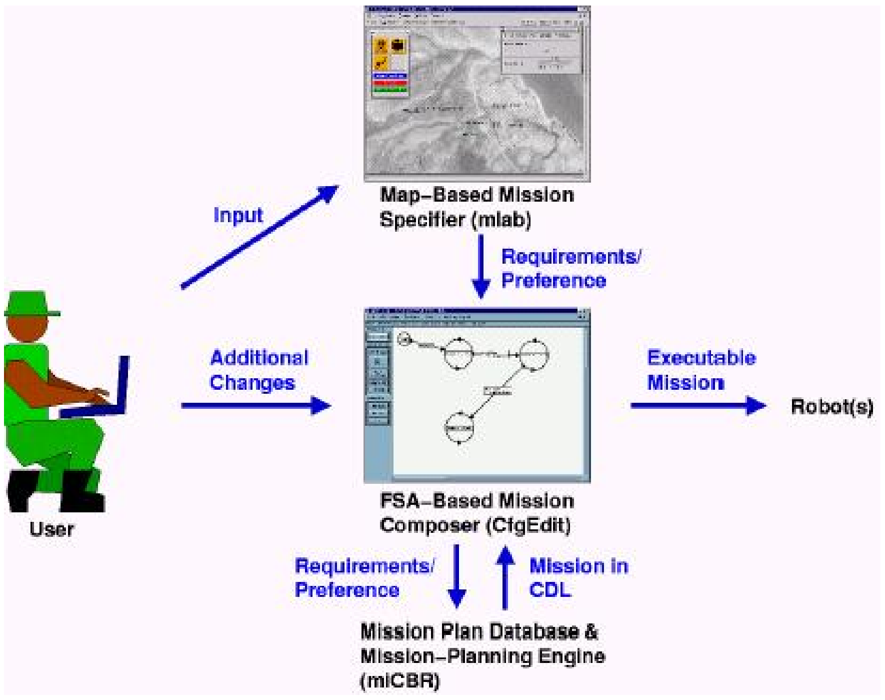
As an example for such interconnection:
A command given by human operator to the robot looks like "turn left" or "move forward".
The human cannot decompose the command to subcommands like "use camera input", when "turn the wheels 5 degrees" and "in case of obstacle stop, take input from camera when ……".
The assembler decomposes the command for the robot to primitive behaviors, and vice versa, combines primitive behaviors in meaningful ways for human.
As we know, an **agent is a software implementation of a behavior**.
So, thanks to assemblage, various multiagent (multibehavior) missions are demonstrated in simulation, like in MissionLab.

MissionLab

MissionLab toolset was developed at Georgia Tech, and grew from the AuRa system.
In *MissionLab*, the human replaces the deliberative system by creating a suitable **behavioral assemblage** which completes the desired task.

The user creates and executes the robot mission plans through its graphical user interface.
A mission-planning “wizard” has been recently integrated into *MissionLab*.
The user composes a mission plan using a set of finite state acceptors (FSAs). The user- specified plan consists of tasks (behavioral states) and perceptual triggers (transition functions). A task is an assemblage of behaviors that the robot executes, and a perceptual trigger is a condition that, when true, causes a state transition to the subsequent task. The user creates a mission by choosing an appropriate sequencing of tasks by adding perceptual triggers to control the order of execution.

Based on the requirements and preferences, the mission-planning engine then suggests a plan from stored past successful mission plans within the database. The user can modify the suggested FSA-based plan if so desired. The resulting mission plan can then be compiled, and executed on a simulated or real robot.


Map–Based Mission Specifier (mlab)
Input
Requirements/ Preference
Additional Changes
Executable Mission
Robot(s)
User
FSA–Based Mission Composer (CfgEdit)
Requirements/ Preference
Mission in CDL
Mission Plan Database & Mission–Planning Engine (miCBR)

Chapter 5

Some AuRa robots and their pictures

Many robots have been instantiated in the AuRa tradition. They implement varying degrees of the AuRA philosophy, from the lowest level of motor schema control to full integration with deliberative reasoning.
A wide range of sensors have been used within AuRA to provide perception for the robots. These sensors include:

- Ultrasound for obstacle avoidance and object recognition
- Computer vision for goal and / or path recognition.
- Laser bar code reader for localization.
- Infrared proximity sensors for object detection and avoidance behaviors.
- Shaft encoders for position estimation.

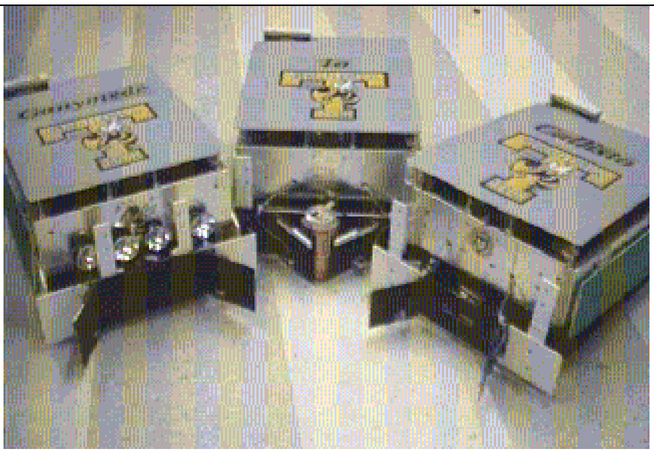
HARV was one of the first robots manufactured by Denning Mobile Robotics and delivered to University of Massachusetts. It is a kinematically holonomic robot which can turn and move in any compass direction. This robot was controlled remotely either via a wire tether or a radio link from Digital Vax computer. HARV was the testbed upon which the first implementations of motor schemas were investigated.

George the first robot acquired at Georgia Tech when the AuRA project relocated from the University of Massachusetts. It is a Denning DRV-1 robot similar to HARV. New motor schemas, including a docking behavior for use manufacturing environments, escape, dodge and an avoid-past behavior for reactive navigation out of box-canyons were initially implemented and tested on George.

Buzz is a Denning MRV-3 that completed in the 1992 AAAI Mobile Robot Competition.
The task for the competition was to explore an office environment and find interesting objects. These objects were free-standing PVC pipes marked with retro-reflective tape so that robot could more easily detect them using computer vision. Buzz uses previously developed motor schemas to seek out and move towards the poles. A novel perceptual schema used visual cues to estimate the location of the poles based on the inherent perspective in the sequencing plans, Buzz extended this capability to more complex situations.

Ren and **Stimpy** are Denning MRV-2 mobile robots equipped with ultrasonic range detectors and positional shaft encoders.
They recently used to evaluate multiagent communication strategies. FSA encodings of behavioral states for a foraging task were ported to Ren and Stimpy then tested with three different communication modes. The tests validated simulation experiments that measured the quantitative differences in performance between the various types of communication. For these tests the robots were linked to off board computers. Ren and Stimpy being upgrade now with on-board computers for future multiagent research.

Stimpy

A similar approach was used in the design of a multi-robot trash-collecting team: the robots, **Io, Ganymede** and **Callisto**. The forage behavior was adapted for the three of robots.	

Chapter 6

Fields of use of AuRA

I find it very important to see examples of use of AuRa.
Because everything we do, we do it with some purpose, it must achieve some goals. But while looking for the examples in the books and Net, unfortunately, I didn't find many of them.
It seems to me that AuRA is not so useable today and probably, its most achievement is that AuRA is an extension of studying robot architectures and on its base other robot architectures have grown up.
Other robotic architectures are in use and developing for today, such as Real-Time Control.

1. USAR: Urban search and rescue

One of the examples that I found, was of Robin Murphy ("Introduction to AI Robotics", MIT Press 2000) of using AuRa in USAR.

Example:
Worker places robot at entrance to unstable building, loads in the floor plan, contextual knowledge and tells the robot to look for survivors efficiently and map out safe routes for workers to pass through. Contextual knowledge includes probability of where people are more likely to be.

How AuRa architecture will do USAR task?

• Cartographer accepts the map
• Navigator uses path planning algorithm to visit nodes in order of likelihood of survivors
• Pilot determines the list of behaviors, Motor Schema Manager instantiates them (MS & PS) and waits for termination
• Homeostatic might notice that robot is running out of power, so opportunistically picks up low probability room on way back to home

But this example is problematic, because it's nearly impossible to use the map in destruction and disaster. Small changes in environment always will take place. This makes Spatial Reasoner unusable and, as a result, all the architecture won't work. Only with the help of human operator it probably may be done (with MissionLab).
The problem is that the majority of tests were executing only in simulation to date.

2. Many robots have been instantiated in the AuRa tradition.

They implement varying degrees of the AuRA philosophy, from the lowest level of motor schema control to full integration with deliberative reasoning. (A.Arkin, AuRA – Principles and Practice in review)

Various architectural components have been applied in a variety of domains including:
Manufacturing environment, Three Dimensional navigation as found in aerial or undersea domains, Indoor and outdoor navigation, Robot competitions, Vacuuming, Military scenarios, Mobile Manipulation, Multi-robot teams

As I've seen, the robots are constructed to show that AuRA is working. But I have not seen examples for AuRA robots that do some other job, like robots that are used in military, or NASA or something else.

3. Developing new architectures and components on base of AuRA

For example:
1. GLUE - A Component Connecting Schema-based Reactive to Higher-level Deliberative Layers for Autonomous Agents (see preferences).
It is a new idea of connecting reactive and deliberative layers in hybrid architecture, and it is based on AuRa and Atlantis architectures.

2. Some Vision and navigation ideas are based on AuRA architecture.
See the "AuRa: an architecture for a vision-based robot navigation" by R.Arkin, E.Riseman and A.Hanson.

3. Researches in intelligent architectures and theories of mind.
See, for example, "Embodiments of Theories of mind " by Petros A.M.Gelepithis.

4. It is possible to find more and more examples…

Chapter 7

Advantages and disadvantages of AuRA
Conclusions

Strengths

Modularity, flexibility, generalizability and hybridization constitute the principal strengths of the AuRA. The value of each of these aspects has been demonstrated in practice both in simulation and on real robotic system.

1. Modularity
AuRA is highly modular by design. Components of the architecture can be replaced with others in a straightforward manner. This is particularly useful in research. Some examples, based on recent or ongoing dissertations, include:

●A specialized **Mission Planner** was developed for an assembly task
This planner was ported to a Denning mobile robot that completed in the 1993 American Association for Artificial Intelligence (AAAI) mobile robot Contest. The planner was further extended in to reason over more general planning tasks.

●The original **A* Spatial Reasoner** has been replaced with Router, a multi-strategy planner. Router models navigable routes as links between nodes instead of the meadow-map representation used previously. The system was tested on Denning mobile robot which successfully navigated from room to room and down corridors.

●**Perceptual schemas** have been expanded to incorporate specialized **action-oriented sensor fusion methods**.
Multiple sensor sources are better in many cases than individual ones; specialized strategies were developed to fuse data together within the context of action-oriented perception.
Dempster-Shafer statistical methods provide the basis for evidential reasoning.

●The original rule-based **Plan Sequencer** has been replaced with a **temporal sequencer** that traverses a finite state acceptor (FSA) expression of a plan.
Each state of FSA represents a specific combination of behaviors that accomplish one step of the task. Transitions are made from one state to another when significant perceptual events trigger them.

2. Flexibility
Another strength of AuRA is the flexibility it provides for introducing **adaptation and learning methods**. In early implementation of AuRA learning arose only from short-term memory of spatial information used by dynamic replanning. Since then, a variety of learning techniques have been introduced including:

● On-line adaptation of motor behaviors using a rule-based methodology.

● Case-based reasoning methods to provide discontinuous switching of behaviors based upon the recognition of new situation.

● Genetic algorithms that configure the initial control system parameters in an efficient manner and that allow a robot to evolve towards its ecological niche in a given task-environment.

3. Generalizability

The generalizability of AuRA to a wide range of problems is another strength.
Various architectural components have been applied in a variety of domains including:

●Manufacturing environment

●Three Dimensional navigation as found in aerial or undersea domains

●Indoor and outdoor navigation

●Robot competitions

●Vacuuming

●Military scenarios

●Mobile Manipulation

●Multi-robot teams

4. Hybridization

Finally, one of the major strengths of AuRA results from the power of wedding two distinct AI paradigms: deliberation and reactivity. The advantages of this strategy have been demonstrated in several other hybrid architectures.

AuRA provides a framework for the conduct of a wide range of robotic research including deliberative planning, reactive control, homeostasis, action-oriented perception and machine learning. It has been motivated but not constrained by biological studies, drawing insight whenever available as a guideline for system design.

Disadvantages of AuRA and proposes to improvement:

1. Deliberative extensions are not well integrated in the system

This consequence I've found in the paper "**GLUE - A Component Connecting Schema-based Reactive to Higher-level Deliberative Layers for Autonomous Agents"** by James Kramer Matthias Scheutz (University of Notre Dame)

As we told, AuRA used a **rule-based system** to control behavior selection. This was replaced by a **finite state machine**. **Both solutions work by engaging or disengaging particular behaviors (i.e., sets of motor schemas) based on perceptual input.**

For example, the **`escape`** behavior, defined as “evade intelligent predators”, will not be activated unless and until an “intelligent predator” is sensed.

Mr. James Kramer Matthias Scheutz claims that the solution of switching on or off whole sets of behaviors is a fairly general way of connecting higher layers to a lower-level layer.
And that the fact that the deliberative extensions use whole sets of motor schemas or controllers also shows that such deliberative extensions are not well integrated in the system. If they were truly integrated, they would not have to reconfigure the reactive layer in such a holistic (global) manner.

As a consequence, such hybrid architectures do not really address the kind of interplay between deliberative and reactive control, that has been part of the motivation for looking at hybrid architectures in the first place (e.g., the integration of a deliberative planner with a reactive plan execution mechanism).
In the case of a deliberative planner, for example, the worst case scenario may require different sets of schemas (i.e., configurations and combinations of motor schemas) for every single plan step, which, in turn, will almost completely remove the autonomy of the lower level (besides the computational overhead of maintaining all the different configurations of motor schemas). Or to put it differently: lower reactive layers “degenerate” to mere implementations of higher level “actions” (similar to action sequencing performed in contention scheduling (Norman & Shallice 1980; Cooper & Shallice 2000)).

In order to retain in part the autonomy of lower levels and to minimize the computational overhead and modifications of the lower reactive layers required for an integration of higher deliberative layers, an alternative option of integrating schema-based reactive systems with deliberative extensions is investigated: ***the modification of outputs from perceptual schemas.***
This modification is done by:
Data transformation – it needs to be transformed into a force vector representation, and
Data maintenance – it needs to be provided and maintained at a time frame appropriate to the reactive layer.

The explanation of the idea can be found in his paper we mentioned above.

2. The limitation by the Spatial Reasoner
The architecture can act only in a known place with stored in the memory map.
I personally think that it is a disadvantage, because it is difficult to find today static environments.

3. Robots can act only nearly in real-time
“AuRA ‘s robots can act **nearly** in real-time, thanks to integrating of motor schemas and perceptual schemas”.
From one side it is an advantage;
from other side, in comparison with today’s Real-Time Control architecture, it is a disadvantage.

Conclusions

AuRA's strengths lie in its

modularity, which permits ready integration of new approaches to various architectural components (it is possibly to add or remove motor schemas, perceptual schemas, sensors and human operator…);

flexibility, as evidenced by the ease of introduction of various learning methodologies and novel behaviors;
generalizability, demonstrated by its applicability to a wide range of domains, including robot competitions, among others; and most importantly,

use of hybridization to exploit the strengths of both symbolic reason and reactive control.

My opinion is that AuRA is a powerful architecture that allows completing different tasks. And in combination with human operator and vision it may be very useful, because the robot can act nearly in real-time, thanks to integrating of motor schemas and perceptual schemas.

But the common problem of nearly all the architectures today is that the implementation costs much money, and in majority of cases is not profitable. This is the main reason of comparatively slow progress in developing and improvement of architectures.
However, there is progress and improvements in this field.

About Ronald C. Arkin

Ronald C. Arkin received the **B.S. Degree** from the **University of Michigan**,
the **M.S. Degree** from **Stevens Institute of Technology**,
and a **Ph.D.** in Computer Science from the **University of Massachusetts**, Amherst in 1987.
In 1987, he joined the College of Computing at the **Georgia Institute of Technology** where he now holds the rank of Professor and is the Director of the Mobile Robot Laboratory.

Dr. Arkin's research interests include reactive control and action-oriented perception for the navigation of mobile robots and unmanned aerial vehicles, robot survivability, multi-agent robotic systems, and learning in autonomous systems. He has over 90 technical publications in these areas.

Funding sources have included the National Science Foundation, ARPA, U.S. Army, Savannah River Technology Center, and the Office of Naval Research. Dr. Arkin is an Associate Editor for IEEE Expert and a member of the Editorial Board of Autonomous Robots. He is a Senior Member of the IEEE, and a member of AAAI and ACM.

References

[Nwana 96] Nwana, H, S 1996: Software Agents: an overview. Knowledge Engineering Review 11 pp11-40, Sept 1996.

[1] AuRA – Principles and Practice in review
(Ronald C. Arkin and Tucker Balch**; 1997**)

(Can be found at http://www.cc.gatech.edu/ai/robot-lab/publications.html)

[2] AuRA: An architecture for vision-based robot navigation
(Ronald C. Arkin, Edward R. Riseman, Allan R. Hanson; **1987**)

(Can be found at http://www.cc.gatech.edu/ai/robot-lab/publications.html)
(Request by email)

[3] Homeostatic Control for a mobile robot: Dynamic replanning in hazardous Environments
(Ronald C. Arkin; **1988**)

(Can be found at http://www.cc.gatech.edu/ai/robot-lab/publications.html)
(Request by email)

[4] **Usability Evaluation of High-Level User Assistance for Robot Mission Specification**
(Yoichiro Endo, Douglas C. MacKenzie, and Ronald C.Arkin; **2002)**

(Can be found at http://www.cc.gatech.edu/ai/robot-lab/publications.html)

[5] **A Hybrid Mobile Robot Architecture with Integrated Planning and Control**
(Kian Hsiang Low, Wee Kheng Leow, Marcelo H. Ang Jr.)
(Can be found at http://www.comp.nus.edu.sg/~ieslkh/pubs/aamas2002.pdf)

[6] GLUE - A Component Connecting Schema-based Reactive to Higher-level
(James Kramer, Matthias Scheutz)
(Can be found at http://www.nd.edu/~airolab/publications/FLAIRS5B03JKramer.pdf)

[7] Motor Schema Based Navigator for a Mobile Robot:
An Approach to Programming by Behavior (Ronald C. Arkin)
(Can be found on http://www.cc.gatech.edu/ai/robot-lab/publications.html
Request by email)

[8] "Embodiments of Theories of mind " (Petros A.M.Gelepithis; 1999)

http://technology.kingston.ac.uk/researchT2/CISgroups/CSL/petros/publications.html

[9] "Multi Agent Robots" (Cai Peng)
http://www.wisdom.weizmann.ac.il/~felix/Robotics_2002/project/multi_agent_robots.txt

[10] I used material about MissionLab from Georgia Tech website:
http://www.cc.gatech.edu/ai/robot-lab/research/MissionLab/

[11] I used material from the website of University of Tennessee:
http://www.cs.utk.edu/~parker/Courses/CS594-fall02/Lectures/