# An Architecture for an Autonomous Learning Robot

Brian Tillotson
Boeing Electronics
P.O. Box 24969 M/S 7J-24
Seattle WA 98124-6269

## ABSTRACT

An autonomous learning device must solve the *example bounding* problem, i.e., it must divide the continuous universe into discrete examples from which to learn. We describe an architecture which incorporates an example bounder for learning. The architecture is implemented in the GPAL program. An example run with a real mobile robot shows that the program learns and uses new causal, qualitative, and quantitative relationships.

## INTRODUCTION

A long term goal of AI research is to produce an autonomous machine that acts as a surrogate for human beings. The machine must perceive the physical environment and be able to change it in useful ways. The machine should accept new goals from a human at any time. When the machine does not know how to achieve its goals or has none pending, it should do experiments to fill gaps in its knowledge. This need for self-guided learning is a major obstacle in building a human surrogate machine.

Machine learning research has rarely addressed the requirement we call *example bounding*. In most machine learning work, the events or examples from which the system learns are somehow bounded or defined before they are input to the learning system. This approach is inadequate for an autonomous machine, which cannot generalize unless it somehow divides the continuous universe into discrete examples. Dieterrich and Michalski[3] briefly address the problem in their program SPARC/E, which plays a card game in which a pattern must be learned. The program is initially unsure whether the pattern is based on the last one, two, or three cards played, so it adjusts the temporal boundaries of events until a meaningful pattern can be recognized.

For humans, example bounding often seems to be guided by temporal or spatial proximity.[1] We have designed and tested an example bounder which uses temporal information to group simple events into examples for inductive learning.[5] If a temporal rule is known to link events of two concepts, then that rule is used to guide the quick, accurate formation of examples. If no such rule is known, then examples are formed by temporal clustering.

Here we propose an autonomous learning device architecture which incorporates an example bounder. We describe the overall architecture and examine each of its major components. We briefly describe an implementation of the architecture, and present an example run. Finally, we discuss the significance of this work and some areas for future work. A glossary is appended at the end of the paper.

## OVERVIEW OF PROPOSED ARCHITECTURE

The architecture we propose is illustrated in Figure 1. It has four major components: a planner, a temporal database of events, a body of conceptual knowledge, and a learner. The

planner controls the creation of new events. Primitive events correspond to actuator commands and sensor inputs; derived events are recognized or computed from simple events or other derived events. The event database maintains information about temporal relationships between events. Conceptual knowledge is divided two ways: a concept is either a descriptor or a rule, and either an hypothesis or a theory. The learner incrementally forms new hypotheses by constructive induction, and uses the planner to carry out experiments which verify or refute hypotheses.

The top-level control of the architecture recognizes three situations. First, if there is a pending goal from the user and the system knows how to achieve the goal, then the planner acts to achieve the goal. Second, for a pending goal which the system does not know how to achieve, the learner concentrates its research on concepts related to the goal. Third, with no pending goals, the learner studies concepts which it heuristically determines to be interesting.

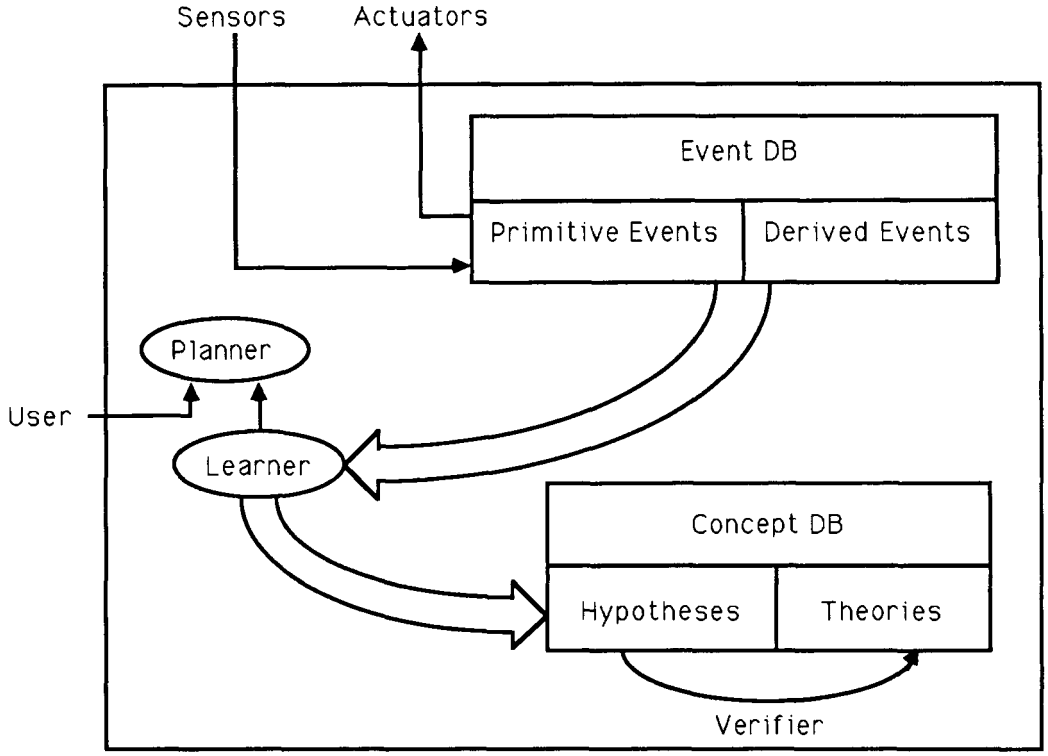


Figure 1. Proposed architecture for autonomous devices.

## The Planner

A goal is input to the planner as a description of a desired event. The concept associated with such an event can have execution knowledge attached to it; the planner uses that knowledge. If no execution knowledge exists, and background knowledge fails to solve the problem, then the planner signals a failure.

## The Temporal Database of Events

The temporal database has some of the features of a time map management system.[2] It maintains a temporal partial order of all events, and can determine the temporal relationship between any two events, including the temporal distance between them.

## Conceptual Knowledge

Conceptual knowledge embodies the system's understanding of the universe. It is divided two ways: a concept is either a descriptor or a rule, and either an hypothesis or a theory.

Descriptor concepts represent attributes, such as temperature or acceleration. Primitive descriptor events are sensor inputs or actuator commands. Values of derived descriptor events are computed from the values of other events; the knowledge of how to do these computations is part of the descriptor concept. Knowledge attached to each descriptor includes the range of observed values.

Rule concepts make predictions in the form, "the presence of a certain type of event implies the presence of another type of event with a specified temporal relationship to the first event". A rule event is a pair of descriptor events which satisfies the rule. Each rule has knowledge of its counter-examples, i.e. events which satisfy the rule's premise but not its conclusion.

A rule hypothesis is a rule which has not been confirmed often enough to be considered reliable, but which is not obviously false. For example, it might have only one observed example and no counter-examples. The verifier knowledge source decides when a rule hypothesis is reliable enough to be a theory, using criteria in the verifier's background knowledge. An example criterion is that the concept must have at least three examples, and the number of counter-examples must be less than 10% of the number of examples. The refuter knowledge source decides when a rule hypothesis is so unreliable that it should be forgotten, e.g. when there are at least three counter-examples, and the number of counter-examples is at least 30% of the number of examples.

A descriptor hypothesis is a descriptor which is not yet known to be useful. A descriptor can be useful in three ways: it can be an observed constant, it can be a term in a rule theory, or it can be a sub-concept of one of these. Our use of observed constants for learning is akin to BACON.[4] The verifier uses background knowledge to decide when a descriptor is a reliable and useful constant. Sub-concepts of constants or rules automatically become theories when the concepts they support become theories.

## The Learner

The conceptual structure of the learner shown in Figure 2 comprises a topic selector, a hypothesis tester, a surprise monitor, and an induction system.

The topic selector decides what theory to investigate. Evaluation criteria include the number of applicable hypotheses and the relationship to unmet goals. The most promising theory becomes the topic. The planner tries to produce an example event, called the topic event.

The hypothesis tester determines whether the topic event acts to support or refute any hypotheses. If supportive, the event and the hypothesis are passed to the verifier, which may promote the hypothesis to a theory; otherwise, they are passed to the refuter.

The surprise monitor notices and tries to explain events that defy expectations. For example, a new event may dramatically extend the observed range of a descriptor. The surprise monitor would search for events which might have caused this change, e.g. the first event of an actuator descriptor. If a plausible cause is found, the surprise monitor forms a rule

hypothesis (and necessary supporting descriptors) to predict such cause-effect pairs in the future.

The major elements of the induction system are the partner selector, the example bounder, and the induction element. The partner selector heuristically chooses a descriptor to be paired with the topic concept. The example bounder pairs individual events of the topic with events of the partner. These event pairs serve as examples for the induction element, which uses constructive generalization rules to find hypotheses relating the topic concept to the partner concept.

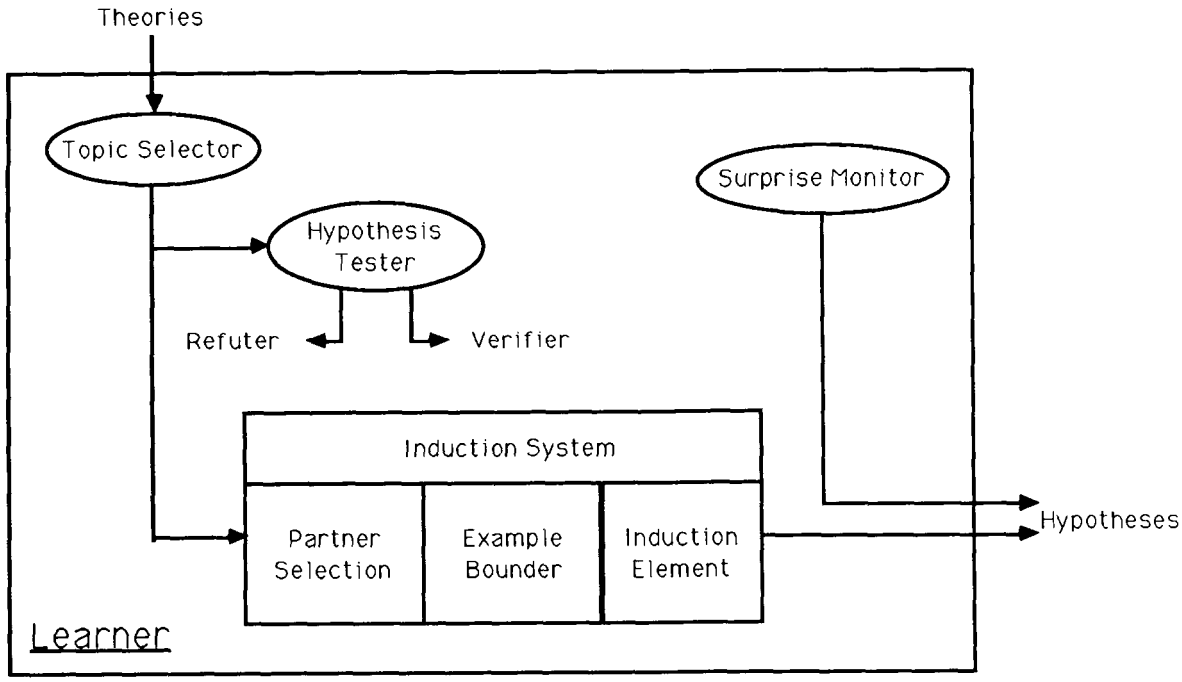


Figure 2. Learner Components

## IMPLEMENTATION AND TEST

We have implemented a simple version of the proposed architecture in the general purpose autonomous learner (GPAL) program. GPAL has been applied to learning about the world of a mobile robot.

GPAL is implemented in Common LISP and remotely controls a Heath/Zenith HERO 2000 mobile robot. Primitive descriptors known to GPAL at start-up are:

1) Send out a SONAR pulse.
2) MOVE the robot forward or backward some distance.
3) Sense the DISTANCE to a wall in front of the robot.

Unknown to GPAL, SONAR events cause DISTANCE events; DISTANCE events cannot be instantiated alone. The robot faces a wall, so the value returned by DISTANCE changes in proportion to the value of MOVE events.

An annotated trace of a GPAL run is listed below. Comments are in italics. Events are shown by name, e.g. EVENT-2071, and by content, e.g. (DISTANCE 90). For clarity and brevity, no individual events are shown after cycle 3.

```
> (gpal)
EVENT-2064 (SONAR)
EVENT-2071 (DISTANCE 90)
"Topic-event: (SONAR)"
"Making rule SONAR-DISTANCE-UNEXPECTED"
"Making rule DISTANCE-SONAR-UNEXPECTED"
End cycle 0
```

*In cycle 0, GPAL studies SONAR. A SONAR event is created, and a DISTANCE event occurs as an unexpected side-effect. GPAL makes two rule hypotheses from this: 1) That a DISTANCE event will always immediately follow a SONAR event, and 2) that a SONAR event will always immediately precede a DISTANCE event.*

```
EVENT-2121 (SONAR)
EVENT-2124 (DISTANCE 91)
"Topic-event: (SONAR) (SONAR)"
"Unexplained events since topic-event: (EVENT-2124)"
End cycle 1

EVENT-2129 (SONAR)
EVENT-2130 (DISTANCE 90.5)
"Topic-event: (SONAR)"
"Verifying SONAR-DISTANCE-UNEXPECTED"
"Verifying DISTANCE-SONAR-UNEXPECTED"
"Inductive partner: DISTANCE"
End cycle 2
```

*In cycles 1 and 2, GPAL tests the hypotheses about SONAR and DISTANCE. In cycle 1, the DISTANCE event is unexplained because its value of 91 is outside the previously known range of [90]. There is no obvious cause for the extension, and the new value is incorporated into the known range of DISTANCE. In cycle 2, enough examples have been gathered to verify the rules relating SONAR and DISTANCE. The hypotheses become theories. GPAL naively tries to induce a relationship between DISTANCE values and SONAR; the example bounder forms examples from event pairs which are examples of the learned rules.*

```
EVENT-2148 (MOVE -20)
EVENT-2149 (SONAR)
EVENT-2150 (DISTANCE 109.5)
"Topic-event: (MOVE -20)"
"Inductive partner: DISTANCE"
"Unexplained events since topic-event: (EVENT-2150)"
"Making descriptor DELTA-DISTANCE"
"Making descriptor ABS-DELTA-DISTANCE"
"Making sub-range SR1/2-ABS-DELTA-DISTANCE"
"Making sub-range SR2/2-ABS-DELTA-DISTANCE"
"Making rule MOVE-SR2/2-ABS-DELTA-DISTANCE-EXCEPTION"
End cycle 3
```

*In cycle 3, GPAL picks MOVE as the topic. It tries to relate the value of MOVE to the value of DISTANCE, so a new DISTANCE event is created (using learned knowledge of how to do this). No temporal rule links MOVE and DISTANCE, so the example bounder pairs events by temporal proximity.*

*The new DISTANCE event has a value far outside the previously observed range. A new operator, MOVE, has been used, so the surprise monitor attributes the range extension to MOVE. New descriptors are created: change in DISTANCE, size of change in DISTANCE, small (SR1/2: x <= 2.0) and large (SR2/2: x >= 2.0) size of change in DISTANCE. These are used in the rule hypothesis that "each large-change-in-DISTANCE event temporally contains a MOVE event".*

```
"Topic-event: (MOVE 4)"
"Inductive partner: DISTANCE"
End cycle 4

"Topic-event: (DISTANCE 105)"
End cycle 5

"Topic-event: (DISTANCE 99.5)"
End cycle 6

"Topic-event: (MOVE -13)"
"Verifying MOVE-SR2/2-ABS-DELTA-DISTANCE-EXCEPTION"
"Verifying SR2/2-ABS-DELTA-DISTANCE"
"Verifying ABS-DELTA-DISTANCE"
"Verifying DELTA-DISTANCE"
"Inductive partner: ABS-DELTA-DISTANCE"
End cycle 7
```

*MOVE and DISTANCE events are generated in cycles 4-7. In cycle 7, the verifier is convinced that the rule linking MOVE to large changes in DISTANCE is true. The rule becomes a theory, which promotes its supporting descriptors to theories.*

*Cycles 8 through 23 are fruitless attempts to find other hypotheses. We resume the example at cycle 24.*

```
"Topic-event: (MOVE -19)"
"Inductive partner: DELTA-DISTANCE"
"Making descriptor DELTA-DISTANCE-/--1.028"
"Making descriptor DELTA-DISTANCE-/--1.028-MINUS-MOVE"
End cycle 24
```

*Here MOVE is inductively matched with DELTA-DISTANCE. The example bounder pairs events of the two descriptors, using the known temporal link between MOVE and SR2/2-ABS-DELTA-DISTANCE. An inductive knowledge source finds a noisy linear relationship between the two. This is represented by new descriptor hypotheses. The first is DELTA-DISTANCE divided by the slope of a DELTA-DISTANCE vs. MOVE graph; the second is this value minus MOVE, which is equivalent to the y-intercept divided by the slope. The second descriptor will be nearly constant if the linearity holds. There are previous examples, but the verifier requires at least one more example after an hypothesis has been proposed.*

```
"Topic-event: (DELTA-DISTANCE -1)"
"Inductive partner: DISTANCE"
End cycle 25

"Topic-event: (SONAR)"
"Inductive partner: SR2/2-ABS-DELTA-DISTANCE"
End cycle 26

"Topic-event: (ABS-DELTA-DISTANCE 0)"
End cycle 27

"Topic-event: (SR2/2-ABS-DELTA-DISTANCE 19.5)"
"Inductive partner: DISTANCE"
End cycle 28

"Topic-event: (MOVE -6)"
"Verifying DELTA-DISTANCE-/--1.028-MINUS-MOVE"
"Verifying DELTA-DISTANCE-/--1.028"
"Inductive partner: DISTANCE"
End cycle 29
```

*In cycle 25, GPAL tries to create a DELTA-DISTANCE-/--1.028-MINUS-MOVE event, but fails; the new descriptor is neither verified nor refuted. In cycle 26, a new example is created, but GPAL is only looking at super-concepts of the topic, SONAR. A similar missed opportunity comes in cycle 28. In cycle 29, GPAL returns its attention to MOVE, makes a new DELTA-DISTANCE-/--1.028-MINUS-MOVE event, and verifies that the descriptor is relatively constant. Both descriptor hypotheses are promoted to theories. Aside from slight numerical adjustments, GPAL now understands everything in its environment.*

## DISCUSSION

We have described the example bounding problem, and claimed that solving this problem is a key to building autonomous machines that learn and are useful. We have proposed a machine architecture which incorporates an example bounder. The architecture has been implemented in the GPAL program.

Tests of the GPAL program with a mobile robot show that it is able to learn the rules governing a simple real environment. The example bounder plays a vital role, forming the examples from which GPAL learns the linear relationship between MOVE and DELTA-DISTANCE. GPAL's gradual learning of the relationship between MOVE and DISTANCE shows a conceptual progression from qualitative physics to quantitative understanding. Early on, GPAL learns that MOVE causes large changes in DISTANCE. Later, it quantifies that relationship. Both sets of knowledge are retained and are available to the planner.

It remains to be shown that the proposed architecture is adequate for general robot learning in complex environments. We will incorporate real-time visual information and two-dimensional motion in the future. Such tests require more heuristics for learning so that GPAL can recognize more subtle regularities in its environment. Greater noise adaptation is particularly important; the heuristics used now sometimes cannot handle the noise of the current environment.

We have barely addressed forgetting, an issue which will gain importance as learning machines with finite memories begin to learn over long times and many domains. Our

architecture addresses only the forgetting of refuted hypotheses. Forgetting of events and of easily re-learned concepts will need to be addressed eventually.

## GLOSSARY

We use the following definitions:

Attribute: a measurable or observable feature, e.g. weight or color.
Complex event: an event comprising two or more simple events, used as input to an inductive learner.
Concept: a rule or a descriptor.
Derived concept: a concept which is defined in terms of one or more other concepts.
Instantiation: creation of the data object which represents a simple event of a specified concept.
Primitive: a descriptor defined by the machine in which the system runs. The instantiation side effects of primitives are the means by which the device interacts with the world.
Simple event: a single observation of one attribute, or an example illustrating a law.
Sub-concept: one of the concepts from which a derived concept was derived. A primitive has no sub-concepts.
Super-concept: a derived concept is a super-concept of concepts from which it is derived.

## REFERENCES

1. Anderson, J.R., "Causal Analysis and Inductive Learning", Proc. of 4th International Workshop on Machine Learning, 1987, 288-299.

2. Dean, T.L., and D.V. McDermott, "Temporal Database Management", Artificial Intelligence 32, 1987, 1-56.

3. Dieterrich, T.G., and R.S. Michalski, "Discovering Patterns in Sequences of Events", Artificial Intelligence 25, 1985, 288-299.

4. Langley, P, G.L. Bradshaw, and H.A. Simon, "Rediscovering Chemistry with the BACON System", in Machine Learning, R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, eds., Tioga, Palo Alto, 1982.

5. Tillotson, B., C. Lin, and J. Bezdek, "Creating Input Sets for Inductive Learning from Simple Events", Proc. of SPIE Applications of Artificial Intelligence VI, 1988, 273-277.