What Sensing Tells Us:
Towards A Formal Theory of Testing for Dynamical Systems

Sheila A. McIlraith
Knowledge Systems Laboratory
Department of Computer Science
Stanford University
Stanford, CA 94305-9020
sam@ksl.stanford.edu

Richard Scherl
Department of Computer Science
New Jersey Institute of Technology
University Heights
Newark, NJ 07102-1982
scherl@cis.njit.edu

Abstract
Just as actions can have indirect effects on the state of the world, so too can sensing actions have indirect effects on an agent's state of knowledge. In this paper, we investigate "what sensing actions tell us"; i.e., what an agent comes to know indirectly from the outcome of a sensing action, given knowledge of its actions and state constraints that hold in the world. To this end, we propose a formalization of the notion of testing within a dialect of the situation calculus that includes knowledge and sensing actions. Realizing this formalization requires addressing the ramification problem for sensing actions. We formalize simple tests as sensing actions. Complex tests are expressed in the logic programming language Golog. We examine what it means to perform a test, and how the outcome of a test affects an agent's state of knowledge. Finally, we propose automated reasoning techniques for test generation and complex-test verification, under certain restrictions. The work presented in this paper is relevant to a number of application domains including diagnostic problem solving, natural language understanding, plan recognition, and active vision.

Introduction
Agents equipped with perceptual capabilities must operate in a world that is only partially observable. To determine properties of the world that are not directly observable, an agent must use its knowledge of the relationship between objects in the world, and its limited perceptual capabilities to infer such unobservable properties. For example, if an agent performs a sense action and observes that there is steam coming out of an electric kettle, then the direct effect of that sensing action is that the agent knows there is steam coming out of the kettle. With appropriate knowledge of the functioning of kettles, the agent should also know that the electrical outlet has power, that the kettle is functioning, and that there is hot liquid inside the kettle – all as indirect effects of the sensing action. Similarly, if the agent wishes to know whether there is power at an electrical outlet, but cannot directly sense this property of the world, the agent may potentially acquire this knowledge by attempting to boil water in a kettle plugged into this outlet.

Copyright © 2000, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.
model the relationship between objects in the world, adopting the associated solution to the ramification problem for world-altering actions. We show that this solution extends to solve the ramification problem in the presence of sensing actions. Next, we define the notion of a test — how to design them and what knowledge can be drawn from their outcomes. In the formalization, simple tests comprise a set of initial conditions and a primitive sensing action. Complex tests are expressed as complex actions in the logic programming language Golog. We examine what it means to perform a test, and how the outcome of a test affects an agent's state of knowledge. Additionally, we examine the issue of selecting tests to confirm, refute, or discriminate a space of hypotheses.

Finally, we investigate the automation of reasoning about tests. We show that regression may be used to verify objective achievement for complex tests written in a subset of Golog. Further restrictions on the form of the complex tests allows the same regression operators to serve as the basis for a simple regression-style planner that generates tests to increase an agent's knowledge with respect to a space of hypotheses.

**Situation Calculus**

The situation calculus language we use, following (Reiter 2000), is a first-order language for representing dynamically changing worlds in which all of the changes are the direct result of named actions performed by some agent, or the indirect result of state constraints. Situations are sequences of actions, evolving from an initial distinguished situation, designated by the constant $S_0$. If $a$ is an action and $s$ a situation, the result of performing $a$ in $s$ is the situation represented by the function $do(a, s)$. Functions and relations whose truth values vary from situation to situation, called fluents, are denoted by a predicate symbol taking a situation term as the last argument. Note that for the purposes of this paper, we assume that our theory contains no functional fluents. Finally, $Pos(a, s)$ is a distinguished fluent expressing that action $a$ is possible to perform in situation $s$. A situation calculus theory $T$ comprises the following sets of axioms:

- foundational axioms of the situation calculus, $\Sigma$;
- successor state axioms, $D_{SS}$;
- action precondition axioms, $D_{AP}$;
- axioms describing the initial situation, $D_{\Delta 0}$;
- unique names for actions, $D_{una}$;
- domain closure axioms for actions, $D_{doa}$.

Successor state axioms, originally proposed by (Reiter 1991) to address the frame problem and extended by (e.g., (Lin & Reiter 1994; McIlraith 2000)) to address the ramification problem, are created by making a causal interpretation of the ramification constraints and a causal completeness assumption and compiling effect axioms of the form:\(^2\)

\[
\text{Pos}(a, s) \land \gamma^+_T(\bar{x}, a, s) \supset F(\bar{x}, do(a, s)),
\]

\[
\text{Pos}(a, s) \land \gamma^-_T(\bar{x}, a, s) \supset \neg F(\bar{x}, do(a, s)),
\]

and ramification (state) constraints of the form:

\[
v^+_T(\bar{x}, s) \supset F(\bar{x}, s),
\]

\[
v^-_T(\bar{x}, s) \supset \neg F(\bar{x}, s),
\]

into Intermediate Successor State Axioms of the form:

\[
\text{Pos}(a, s) \supset [F(\bar{x}, do(a, s))] \equiv \Phi^+_T,
\]

\[
\Phi^+_T \equiv \gamma^+_T(\bar{x}, a, s) \lor v^+_T(\bar{x}, do(a, s)) \\
\lor (F(\bar{x}, s) \\
\land \neg (\gamma^+_T(\bar{x}, a, s) \lor v^+_T(\bar{x}, do(a, s))))
\]

I.e., if an action is possible in situation $s$, then it implies that the fluent is true in $do(a, s)$ iff an action made it true - or a state constraint made it true - or it was already true and neither an action nor a state constraint made it false. Such intermediate successor state axioms provide a compact representation of a solution to the ramification problem for a common class of state constraints. (McIlraith 2000) shows that for what are essentially acyclic causal ramification constraints, repeated regression rewriting (e.g., (Reiter 1991)) of $\Phi^+_T, \Phi^+_T, \Phi^+_T$, repeatedly rewrites the ramification constraints that are relativized to $do(a, s)$ in (6) above, and is guaranteed to terminate in a formula whose fluents are relativized to situation $s$ rather than $do(a, s)$. Both the intermediate and the less compact (final) successor state axioms which result from the regression provide closed-form solutions to the frame and ramification problem for the designated class of state constraints.

To illustrate sensing and testing in partially observable environments, we present a partial axiomatization of a car repair domain, derived from The Complete Idiot's Guide to Trouble-Free Car Care (Ramsey 1999). Our domain includes world-altering actions such as turn on(x) and turn off(x), where $x$ is radio or lights. These have the effect that the radio or lights are on/off in the resulting situation. Actions turn(key) and release(key) have the effect that the ignition is begun (turning on), or not, in the resulting situation. These actions are defined in terms of effect axioms and are combined with the following self-explanatory state constraints to provide successor state axioms. For notational convenience we abbreviate: transmission - trans, interlock - interl, solenoid - soln, engine - eng, battery - bat; ignition system - ign, start system - start, empty gas tank - empt,gas_tank.

\[
\text{empt}_i(gas_tank, a) \supset \neg \text{start}_i(a) \quad (7)
\]

\[
\text{on}(intrk, a) \supset \neg \text{start}_i(a) \quad (8)
\]

\[
\text{on}(intrk, a) \supset \neg \text{start}_i(a) \quad (9)
\]

\[
\text{on}(soln, a) \supset \neg \text{start}_i(a) \quad (10)
\]

\[
\text{on}(starter, a) \supset \neg \text{start}_i(a) \quad (11)
\]

\[
\text{on}(trans) \land \neg \text{in} \text{g}_i(t, trans, a) \supset \text{on}(intrk, a) \quad (12)
\]

\[
\text{on}(trans) \land \neg \text{depress}_i(clutch, a) \supset \text{on}(intrk, a) \quad (13)
\]

\[
\text{turning}_i(ign) \land \text{on}(intrk, a) \supset \neg \text{start}_i(eng, a) \quad (14)
\]

\[
\text{turning}_i(ign) \land \text{empt}_i(gas_tank, a) \supset \neg \text{start}_i(eng, a) \quad (15)
\]
Turning light on: \( \neg \text{dark-light}(s) \wedge \neg \text{on}(\text{light}, s) \supset \text{on}(\text{light}, s) \)  
(16)

Opening radio: \( \text{open}(\text{radio}, s) \supset \neg \text{closed}(\text{radio}, s) \)  
(17)

Opening door: \( \text{open}(\text{door}, s) \supset \neg \text{closed}(\text{door}, s) \)  
(18)

Turning off radio: \( \text{on}(\text{radio}, s) \supset \neg \text{off}(\text{radio}, s) \)  
(19)

Space precludes listing all the successor state axioms. There is one (intermediate) successor state axiom for each fluent.

E.g., axioms (7)-(11) compile into intermediate successor state axiom (20):

\[
\begin{align*}
\text{Poss}(a, s) & \supset \text{do}(a, s) \\
& \equiv \\
& \neg \text{empty}(\text{gas tank}, a) \wedge \neg \text{on}(\text{flash light}, a) \\
& \wedge \neg \neg \text{on}(\text{radio}, a) \wedge \neg \text{on}(\text{light}, a) \\
& \wedge \neg \text{on}(\text{do(a), s}) \\
& \wedge \neg \text{on}(\text{do(a), s}) \\
& \wedge \neg \text{on}(\text{do(a), s}) \\
& \wedge \neg \text{on}(\text{do(a), s})
\end{align*}
\]

As described in (McIlraith 2000), the axioms describing the initial situation, \( S_0 \), contain what is known of the initial situation as well as the ramification constraints of the form of (3) and (4), relativized to \( S_0 \).

Knowledge and the Ramification Problem

In (Scherl & Levesque 1993), the situation calculus language without state constraints was extended to incorporate both knowledge and sensing actions. World-altering actions change the state of the world, sensing actions have no effect on the state of the world but rather change the agent's state of knowledge. In our example, sensing actions include check fuel, check car, start, check radio, noise, etc., which have the effect of the agent knowing empty gas tank, \( \text{do}(a, s) \), startable (do(a), s), and noise radio, do(a), s).

The notation \( \text{Knows}(a, s) \) (read as \( \omega \) is known in situation \( s \)), where \( \omega \) arbitrary formula, is an abbreviation for a formula that uses \( K \). For example \( \text{Knows}(\text{on}(\text{block}, \text{block}), s) \) abbreviates: \( \forall \forall (s', s) \supset \text{on}(\text{block}, \text{block}, s') \).

The notation \( K\text{whether}(a, s) \) is an abbreviation for a formula indicating that the truth value of \( \phi \) is known.

\[
K\text{whether}(a, s) \equiv \text{Knows}(a, s) \lor \text{Knows}(\neg a, s)
\]

Following the notation of (Levesque 1996), each sense action \( a \) has a sensed fluent, \( S(a, s) \) associated with it, and for each such \( a \), \( D \) entails a sensed fluent axiom:

\[
S(a, s) \equiv \psi(s),
\]

which says that performing the sense action \( a \) tells the agent whether the formula \( \psi(s) \) is true or false. Thus, \( D \models K\text{whether}(\psi, s, s) \) where \( a \) is an action with a sensed fluent equivalent to \( \psi \).

For the sense action check fuel the sensed fluent axiom is:

\[
S(\text{check fuel}, s) \equiv \text{empty}(\text{gas tank}, s)
\]

which tells us whether or not the gas tank is empty. For world-altering actions, \( D \) entails \( S(a, s) \equiv \text{True} \).

In (Scherl & Levesque 1993), a successor state axiom for the \( K \) fluent is developed. Its form is as follows:

\[
\text{Successor State Axiom for } K
\]

\[
\begin{align*}
\text{Poss}(a, s) & \supset \text{K}(s', s) \\
& \equiv \exists a' \text{Poss}(a, s') \land K(s', s) \\
& (s' = \text{do}(a, s')) \\
& \land
\end{align*}
\]

which says that after doing action \( a \) in situation \( s \), the agent thinks it could be in a situation \( s' \) iff \( s' = \text{do}(a, s') \) and \( s' \) is a situation that was accessible from \( s \), and where \( s \) and \( s' \) agreed on the truth value of \( SF(a, s) \), e.g., the truth value of empty gas tank. Thus, for all situations \( do(a, s) \), the \( K \) relation will be completely determined by the \( K \) relation at \( s \) and the action \( a \). This extends Reiter's solution to the frame problem (without ramifications and without knowledge) to the case of the situation calculus with sensing actions.

**Proposition 1** In the situation calculus theory described above, the agent knows the successor state axioms and the ramification constraints.

This follows from the fact that the successor state axioms are universally quantified over all situations, and the ramification constraints explicitly hold in \( S_0 \) and are entailed in all successor situations, by the successor state axioms.

**Theorem 1** (Correctness of Solution) The proposed solution to the frame and ramification problems for world-altering and sensing actions ensures that knowledge only changes as appropriate, as defined by Theorems 1, 2, 3 (Scherl & Levesque 1993). Furthermore, the agent knows the indirect effects of its sensing actions.

Thus, the successor state axioms for world-altering and sensing actions, together address the frame and ramification problems.

**Testing**

The purpose of a test is to attempt to determine the truth value of certain properties of the world, that may or may not be directly observable. A test is often performed with respect to a set of hypotheses, with the objective of eliminating as many hypotheses as possible from the set of hypotheses being entertained. Testing has been studied extensively for the specific problem of IC circuit testing, but there is little work on testing for rich dynamical systems such as the ones we examine here. The notion of a static test was briefly discussed in (Moore 1985, litmus example), and further developed for static systems in (McIlraith 1994; McIlraith & Reiter 1992). We build directly upon the work in (McIlraith 1994) with the objective of developing a formal theory of testing for dynamical systems.

Informally, a simple test comprises a set of initial conditions that may be established by the agent, together with the specification of a primitive sensing action, which determines what the agent will directly come to know as the result of the test. In our car repair domain, we can test the battery by checking the radio for noise. The initial conditions for such a test might be on radio, s. Then we can perform the sensing action check radio noise to see whether the radio is emitting noise. Note that the precondition for performing the action check radio noise, \( \text{Poss}(\text{check radio noise}, s) \equiv \text{inside}(\text{car}, s) \), is different from the initial conditions of the test. Both must hold and must be consistent with the theory and with the current hypotheses being entertained, in order to execute the test.

We distinguish between two types of tests, **truth tests** which tell us whether the properties being sensed are true in
the physical world, and functional tests, which tell us what values of the properties are true in the physical world. For the purposes of this paper, we restrict our attention to truth tests, and our sensing actions to so-called binary sense actions which establish the truth or falsity of a sensed formula.

Definition 1 (Simple Test)
A simple test is a pair, (I, a), where I, the initial conditions, is a conjunction of literals, and a is a binary sense action whose sensed formula contains no free variables.

(onsradio, s), checkradio_noise) is an example of a simple test, following the discussion above. We now define the notion of a test for a particular hypothesis space, represented by the set HY 'P. We restrict the hypotheses, H(s) \in HY 'P to be conjunctions of fluents whose non-situation terms are constants, and whose situation term is a situation variable s. In our car repair domain, an example hypothesis space might be {ab(batt, s), ab(abtdk, s), empty(gasTank, s)}.

Definition 2 (Test for Hypothesis Space HY 'P)
A test (I, a) is a test for hypothesis space HY 'P in situation s if \( D \land I \land \text{Poss}(a, s) \land H(s) \) is satisfiable for every H(s) \in HY 'P.

That is, the state the world must be in to execute the sensing action must be satisfiable, under the assumption that any one of the hypotheses in the hypothesis space could be true. Consider that D entails the safety constraint \( \neg\text{explosion}(s) \) and the axiom \( \text{spark}(s) \land \text{gasLeak}(s) \supset \text{explosion}(s) \), and that our hypothesis space is \{\text{gasLeak}(s), \text{ab(sparkPlug, s)}\}. A reasonable test for \text{ab(sparkPlug, s)} is to try to create sparks at the plug. Unfortunately such a test would cause an explosion in the presence of a gas leak. The satisfiability check above precludes such a test.

Definition 3 (Confirmation, Refutation)
The outcome \( \alpha \) of the test (I, a) confirms H(s) \in HY 'P if \( D \land I \land \text{Poss}(a, s) \land H(s) \) is satisfiable and \( D \land I \land \text{Poss}(a, s) \land H(s) \land \text{Knows}(H \supset \alpha, s) \) refutes H(s) \iff \( D \land I \land \text{Poss}(a, s) \land H(s) \land \text{Knows}(H \supset \neg\alpha, s) \).

If the outcome of test (onsradio, s), checkradio_noise) is \( \text{noise}(\text{radio}, \text{do(a, s)}) \), then our test refutes the hypothesis \text{ab(batt, s)}, following Axiom (17), and we can eliminate \text{ab(batt, s)} from our hypothesis space, HY 'P.

Observe that a test outcome that refutes an hypothesis H(s) allows us to eliminate it from HY 'P. Unfortunately, a test outcome that confirms an hypothesis is generally of no deterministic value, resulting in no reduction in the space of hypotheses. As we will see in a section to follow, there are exceptions that depend on the criteria by which the hypothesis space is defined.

In the sections to follow we use these basic definitions to define discriminating tests and relevant tests. These tests are distinguished by the effect their outcome will have on a general space of hypotheses.

Discriminating Tests
Notice that in our example above, if we had observed \( \neg\text{noise}(\text{radio}, \text{do(a, s)}) \), then by the definition, this would have confirmed the hypothesis \text{ab(batt, s)}, but it would have been of little value in discriminating our hypothesis space. All hypotheses remain in contention. Discriminating tests are those tests (I, a) that are guaranteed to discriminate an hypothesis space HY 'P, i.e., which will refute at least one hypothesis in HY 'P, regardless of the test outcome.

Definition 4 (Discriminating Tests)
A test (I, a) is a discriminating test for the hypothesis space HY 'P if \( D \land I \land \text{Poss}(a, s) \land H(s) \) is satisfiable for all H(s) \in HY 'P, and there exists H(s), H(s) \in HY 'P such that the outcome \( \alpha \) of test (I, a) refutes either H(s) or H(s), no matter what that outcome might be.

Proposition 2
After we perform a discriminating test, (I, a), Knows(\neg H, s), for some H(s) \in HY 'P.

In general, we would like a discriminating test to refute half of the hypotheses in the hypothesis space, regardless of the test outcome. By definition, a discriminating test must refute at least one hypothesis in the hypothesis space.

Definition 5 (Minimal Discriminating Tests)
A discriminating test (I, a) for the hypothesis space HY 'P is minimal if for no proper subconjunct I' of I is (I', a) a discriminating test for HY 'P.

Minimal discriminating tests preclude unnecessary initial conditions for a test.

In some cases, we are interested in identifying a test that will establish the truth or falsity of a particular hypothesis. An individual discriminating test does precisely this.

Definition 6 (Individual Discriminating Tests)
A test (I, a) is an individual discriminating test for the hypotheses H(s) and \( \neg H(s) \in HY 'P if D \land I \land \text{Poss}(a, s) \land H(s) \) is satisfiable for all H(s) \in HY 'P and the outcome \( \alpha \) of test (I, a) refutes either H(s) or \( \neg H(s) \), no matter what that outcome might be.

Proposition 3
After we perform an individual discriminating test (I, a), Kwether(H, s) for some H, \in HY 'P.

The test \{\text{checkFuel}\} is such a test. The outcome will be one of \( \neg\text{empty(gasTank, do(a, s))} \) or \( \text{empty(gasTank, do(a, s))} \). Thus, as the result of performing checkFuel in the physical world, the agent Kwether(\text{empty(gasTank, s)})

We can similarly define the notion of a minimal individual discriminating test, and a minimal relevant test, below.

Relevant Tests
In the majority of cases we will not be so fortunate as to have discriminating tests. Relevant tests are those tests (I, a) that have the potential to discriminate an hypothesis space HY 'P, but which cannot be guaranteed to do so. Given a particular outcome \( \alpha \), a relevant test may refute a subset of the hypotheses in the hypothesis space HY 'P, but may not refute any hypotheses if \( \neg\alpha \) is observed. Since we can't guarantee the outcome of a test, these tests are not guaranteed to discriminate an hypothesis space. (onsradio, s), checkradio_noise) is an example of such a test.
Definition 7 (Relevant Tests)

A test \((I, a)\) is a relevant test for the hypothesis space \(HYP \iff D \land I \land \text{Poss}(a, s) \land H(s)\) is satisfiable for all \(H(s)\) in \(HYP\), and the outcome \(\alpha\) of test \((I, a)\) either confirms a subset of the hypotheses in \(HYP\) or refutes a subset.

By definition, a relevant test confirms or refutes at least one hypothesis in \(HYP\), and it follows that every discriminating test is a relevant test.

In addition to discriminating and relevant tests, there is a third class of tests. Constraining tests do not refute an hypothesis, regardless of the outcome, but they do provide further knowledge that is relevant to the hypothesis space and which the agent can exploit in combination with other tests. We discuss this notion in a longer paper.

Testing Hypotheses

In the previous section we observed that a test outcome that refutes an hypothesis \(H(s) \in HYP\) allows us to eliminate it from \(HYP\), but that in general an outcome that confirms \(H(s)\) has no value in reducing the hypothesis space. In this section, following (McIlraith 1994), we show that when the hypothesis space is determined using a consistency-based criterion this is indeed true, but when the hypothesis space is defined abductively, confirming test outcomes serve to eliminate those hypotheses that are not confirmed, i.e., that do not explain the test outcome.

Definition 8 (Consistency-Based Hypothesis Space)

A consistency-based hypothesis for \(D\) and outcome \(\alpha\) of the test \((I, a)\) is any \(H(s) \in HYP\) such that \(D \land I \land \text{Poss}(a, s) \land H(s) \land \alpha\) is satisfiable.

Proposition 4 (Eliminating C-B Hypotheses)

The outcome \(\alpha\) of a test \((I, a)\) eliminates those consistency-based hypotheses, \(H(s) \in HYP\) that are refuted by test outcome \(\alpha\).

Definition 9 (Abductive Hypothesis Space)

An abductive hypothesis for \(D\) and outcome \(\alpha\) of the test \((I, a)\) is any \(H(s) \in HYP\) such that \(D \land I \land \text{Poss}(a, s) \land H(s)\) is satisfiable, and \(D \land I \land \text{Poss}(a, s) \land H(s) \models \alpha\).

Proposition 5 (Eliminating Abductive Hypotheses)

The outcome \(\alpha\) of a test \((I, a)\) eliminates those abductive hypotheses, \(H(s) \in HYP\) that are not confirmed by test outcome \(\alpha\).

Thus, in the case of abductive hypotheses, unlike consistency-based hypotheses, both confirming and refuting test outcomes have the potential to eliminate hypotheses.

Proposition 6 (Efficacy of Tests)

Any outcome \(\alpha\) of a relevant test \((I, a)\) can eliminate abductive hypotheses, whereas only a refuting outcome can eliminate consistency-based hypotheses. Discriminatory test outcomes, by definition, can eliminate either consistency-based or abductive hypotheses, regardless of the outcome.

Complex Tests

In the previous section, we defined the notion of a simple test \((I, a)\), and characterized the circumstances under which the outcome of such a test would discriminate an hypothesis space. Indeed, to discriminate an hypothesis space, we may need a sequence of simple tests, interleaved with world-altering actions in order to achieve the initial conditions for a test. Likewise, the selection and sequencing of sensing and world-altering actions may be conditioned on the outcome of previous sensing actions. In the section to follow, we examine the problem of generating tests using regression. As we will see, generating tests, especially tests that involve sequences of sensing and world-altering actions is hard. In many instances, we need not resort to computation. The domain axiomatizer can articulate procedures for testing aspects of a system, just as the author of The Idiot's Guide has done in the domain of car repair. The logic programming language, Golog (a GOL in LOGic) (Levesque et al. 1997) provides a compelling language for specifying such tests, as we describe briefly here.

Only a sketch of Golog is given here. See (Levesque et al. 1997) for a full discussion of the language and also a Prolog interpreter. Golog provides a set of extralogical constructs (such as action sequencing, if-then-else, while loops) for assembling primitive actions, defined in the situation calculus, into macros that can be viewed as complex actions. The macros are defined through the predicate Do\((\delta, s, s')\) where \(\delta\) is a complex action expression. Do\((\delta, s, s')\) is intended to mean that the agent's doing action \(\delta\) in situation \(s\) leads to (a not necessarily unique) situation \(s'\). The inductive definition of Do includes the following cases:

- \(\text{Do}(a, s, s')\) — simple actions
- \(\text{Do}(c?, s, s')\) — tests (referred to as G-tests in this paper)
- \(\text{Do}([\delta_1; \delta_2], s, s')\) — sequences
- \(\text{Do}(\delta_1; \delta_2, s, s')\) — nondeterministic choice of actions
- \(\text{Do}((\Pi_\alpha)\delta, s, s')\) — nondeterministic choice of parameters
- \(\text{Do}(\text{if } \phi \text{ then } \delta, \text{ else } \delta', s, s')\) — conditionals, where we restrict \(\phi\) to a G-test

Do\((\text{while } \phi \text{ do } \delta, s, s')\) — while loops

Space does not permit giving the full expansion for each of the constructs, but they can be found in (Levesque et al. 1997). The only change here is that the definition of the G-test construct (including the implicit G-test in the condition construct) must expand into a G-test involving knowledge.

The following is a partial example of a complex test written in Golog, and derived from (Ramsey 1999). This particular procedure is designed to help discriminate the space of hypotheses generated when a car won't start, namely \(\{\text{ab(intric, s), empty(gas, tank, s), ab(batt, s), ab(solud, s), ab(ign, wires, s), ab(start, s)}\}\). In a diagnostic application such as this one, Golog procedures may also be written to combine testing with repair.

**proc CarWon'tStart**

\[
\text{if} \sim \text{startable} \text{then} \text{checkInterlock;}
\]

\[\text{We are taking the simplest approach towards incorporating sensing actions into Golog. All actions are on-line. In other words, they are executed immediately without any possibility of backtracking. Other options for completely off-line execution (Lakemeyer 1999) and a mixture of off-line and on-line execution (De Giacomo & Levesque 1999a) have been discussed in the literature.} \]
If (~ AB(INTRKL)) then CHECK.GAS.TANK;
If (~ AB(INTRKL)) then CHECK.GAS.TANK;
If (~ AB(BATT)) then CHECK.BATTERY;
If (~ AB(BATT)) then CHECK.BATTERY;
If (~ AB(SOLO)) then CHECKIGN WIRES;
If (~ AB(IGN WIRES)) then CHECKIGN WIRES;
If (~ AB(BATTER)) then CHECKIGN WIRES;
If (~ AB(IGN WIRES)) then CHECKIGN WIRES;
end if
end if
end if
end if
end if
end if
end if
end if



proc CHECKBATTERY

then CHECK.BATTERY

endproc

Observe that complex tests often involve world-altering actions which serve to establish the preconditions and initial conditions for embedded simple tests. Also observe that in achieving the preconditions or initial conditions for simple tests, these actions change the state of the world, including potentially changing the space of hypotheses. For example, if a flashlight isn’t emitting light, and one hypothesis is that the batteries are dead, a good way to test this is to replace them with fresh batteries, and see whether the flashlight then works. However, replacing the flashlight batteries potentially changes the state of one of the hypotheses.

In diagnosis domains, such as the ones above, it is often desirable to combine fault detection (hypothesis testing) with repair and to take actions to eradicate faults as easily as to diagnose them (McIlraith 1997; Baral, McIlraith, & Tran 2000). However, in cases where it is desirable not to alter the truth status of the hypothesis space, care must be taken to design and verify and/or generate tests that maintain designated knowledge constraints and world constraints. E.g., we don’t want to determine whether the gas tank is empty by draining it!

Automated Reasoning About Tests

In the previous section we introduced the notion of a complex test, demonstrating that such tests could sometimes be specified in Golog. In this final technical section we briefly examine the use of automated reasoning techniques, and in particular the use of regression rewriting, for the purpose of verifying certain properties of Golog-specified complex tests, and for generating complex tests as conditional plans. Our presentation draws upon (Lepersague 1994) and (Reiter 2000). Related approaches to conditional planning include (Rosenschein 1981; Manna & Waldinger 1987; Lobo 1998).

Consider the Golog complex test given above to help discriminate the space of hypotheses generated when a car won’t start. To verify that it is an individual discriminating test, it is necessary to ensure that for at least one of the hypotheses $H$, Knows($H, s$) holds, where $s$ is the situation resulting from the execution of the Golog procedure, i.e. Do( CarWontStart, $S_0$, $s$). Thus, we would like to be able to entail $\forall H \in RV, \text{Knows}(H, s)$, and in particular $\text{Knows}(\text{empty}(\text{gas}.\text{tank}), s)$, for example. A verification that the procedure is a discriminating test would involve ensuring that for at least one $H$, Knows($\neg H, s$) holds in the final situation, i.e. $V_{\text{final}} \text{Knows}(\neg H, s)$.

In (Scherl & Levesque 1993), a form of regression (based on the discussion in (Reiter 1991)) is developed for the situation calculus with sensing actions. Through the application of regression, reasoning about situations reduces to reasoning in the initial situation, $S_0$. Given a ground situation term $s$ (i.e. a term built on $S_0$ with the function do and ground action terms $s_a$) the problem is to determine whether the axiomatization of the domain $D$ entails $G(s_a)$ where $G$ (the intended objective of the procedure) is an arbitrary sentence including knowledge operators. This question is reduced to the question of whether or not the axiomatization of the initial situation entails the regression of $G(s_a)$, i.e. $\mathcal{G}(G(s_a))$. Since the result of regression is a formula in an ordinary modal logic of knowledge (i.e. a formula without action terms and where the only situation term is $S_0$) an ordinary modal theorem proving method may be used to determine whether or not the regressed formula is entailed by the axiomatization of the initial situation, $D_{S_0}$. In our case $G$ will be a formula made up of subformulas of the form $\text{Knows}(H, s)$ or $\text{Knows}(\neg H, s)$, where $H$ is an hypothesis.

The regression operator $\mathcal{R}$ is defined relative to a set of successor state axioms $P_{ss}$. The first four parts of the definition of the regression operator $\mathcal{R}$ concern world-altering actions and are taken from (Reiter 2000).

1. When $W$ is a non-fluent atom, including equality atoms, and atoms with the predicate symbol Exor, or when $W$ is a fluent atom or Knows operator, whose situation argument is the situation constant $S_0$, $\mathcal{R}[W] = W$.

2. When $F$ is a relational fluent (other than $K$) whose successor state axiom in $P_{ss}$ is

$$\text{Poss}(a, s) \supset \{F(x_1, \ldots, x_n, do(a, s)) \equiv \Phi_F\}$$

then

$$\mathcal{R}(F(t_1, \ldots, t_n, do(a, s))) \equiv \Phi_F$$

3. Whenever $W$ is a formula,

$$\mathcal{R}[\neg W] = \neg \mathcal{R}[W],$$

$$\mathcal{R}[\forall W] = (\forall W) \mathcal{R}[W],$$

$$\mathcal{R}[\exists W] = (\exists W) \mathcal{R}[W].$$

4. Whenever $W_1$ and $W_2$ are formulas,

$$\mathcal{R}[W_1 \land W_2] = \mathcal{R}[W_1] \land \mathcal{R}[W_2],$$

$$\mathcal{R}[W_1 \lor W_2] = \mathcal{R}[W_1] \lor \mathcal{R}[W_2],$$

$$\mathcal{R}[W_1 \Rightarrow W_2] = \mathcal{R}[W_1] \Rightarrow \mathcal{R}[W_2].$$

Following (Scherl & Levesque 1993), additional steps are needed to extend the regression operator to sensing actions.

Two definitions are needed for the specification to follow. When $\varphi$ is an arbitrary sentence and $s$ a situation term, then $\varphi[s]$ is the sentence that results from adding an extra argument to every fluent of $\varphi$ and inserting $s$ into that argument

$$\mathcal{R}$$

Some details are omitted here (e.g. regression of functional fluents, and the equality predicate). Also note that the formula to be regressed must be regressive. This concept is fully defined in (Reiter 2000).

Regression of sensing actions that make known the denotation of a term (e.g. an action of reading a number on a piece of paper) is not discussed here.
position. The reverse operation $\varphi^{-1}$ is the result of removing the last argument position from all the fluent in $\varphi$.

Step v covers the case of regressing a world-altering action through the Knows operator. Step vi covers the cases of regressing a sensing action through the Knows operator. In the defintions below, $s'$ is a new situation variable.

v. Whenever $a$ is not a sensing action,
$$R[\text{Knows}(W, \text{do}(a, s))] = \text{Knows}(R[\text{do}(a, s')])^{-1}, s).$$

vi. Whenever $a$ is a sensing action, where $\psi$ is a formula such that $D$ entails that $\psi[s]$ is equivalent to $SF(a, s)$,
$$R[\text{Knows}(W, \text{do}(a, s))] = ((\psi(s) \supset \text{Knows}(\psi, s) \supset R[\text{do}(a, s')]^{-1}, s)) \land \neg \psi(s).$$

An additional operator $C$ needs to be defined to handle the expansion of the complex actions found in Golog, so that we can apply regression\(^7\). We are only considering a subset of Golog programs – those composed of simple actions, sequencings, and conditionals. We also add the empty action $\text{noOp}$ or $[\text{names for the same operation}].$ Also note that $\pi_a(\overline{s}, s)$ stands for the preconditions of $a(\overline{s})$ as specified in the action precondition axiom, $D_{\text{pre}}, \text{Preas}(a(\overline{s}), s) \equiv \pi_a(\overline{s}, s).$

vii. $C(\text{noOp}, W, s) = W(s)$

ix. $C(a(\overline{s}), d_1, W, s) = \pi_a(\overline{s}, s) \land C(a, W, \text{do}(a, s))$, where $a(\overline{s})$ is a ground non-sensing simple action term.

x. $C(\text{if } \phi(\overline{s}) \text{ then } \delta_1 \text{ else } \delta_2, W, s) =$
$$\text{Knows}(\phi(\overline{s}), s) \land \text{Knows}(\delta_1, W, \text{do}(a, s)) \land \text{Knows}(\neg \phi(\overline{s}), s) \land \text{Knows}(\delta_2, W, s).$$

We are assuming that the agent is able\(^4\) to execute the Golog test procedure. In particular, the programmer (of the test procedure) must have ensured that at the point where an $\text{if } \phi(\overline{s}) \text{ then } \delta_1 \text{ else } \delta_2$ statement is encountered, the executing agent must $\text{Knows}(\phi, s)$. If not, the procedure will fail.

In the following theorem (a generalization of Theorem 2 from (Lespérance 1994), recall $R^*(\varphi)$ indicates the repeated regression of $\varphi$ until further applications leave the formula unchanged.

**Theorem 2** For any Golog procedure $\delta$, consisting of simple actions, sequencings, and conditionals, and $G$ an arbitrary closed regreessable formula that may include knowledge operators:
$$D \models \exists n(\text{Do}(\delta, S_n, s) \land G(s))$$
iff
$$D_{\delta_0} \cup D_{\text{pre}} \models R^*(C(\delta, G, S_0)).$$

Theorem 2 shows it may be verified that any Golog testing routine (utilizing concatenation and conditionals) achieves its intended objective $G$ through the use of regression followed by theorem proving in the initial database. The successor state axioms ($D_{\delta_n}$) are only used in the regression procedure. This theorem can be extended to likewise verify other properties of our Golog procedures.

We can use the above regression operator as the basis for a simple conditional planning algorithm for constructing complex tests. Following (Lespérance 1994), we consider only normal form conditional plans. These are conditional plans in which the condition in a conditional (e.g. the $\phi$ in $\text{if } \phi(\overline{s}) \text{ then } \delta_1 \text{ else } \delta_2$) must be a sensed formula. Thus we can require that prior to any conditional with the $G$-test $\omega$, there must be an action $a$ such that $a$ is a sensing action and $D \mid SF(a, s) \equiv \phi(s)$. This guarantees that the program executing the test will always $\text{Knows}(\phi, s)$ when a conditional is encountered. For any complex test (that is executable) consisting only of concatenation and conditionals, there must be an equivalent test in this normal form.

For $i = 1, 2, 3, \ldots$, we can define the sentences $F_1$ as:
$$\Gamma_i = \text{C}(G(s))$$

$$\Gamma_i \models \exists n(\text{Do}(\delta, G_n, s) \land G(s))$$
iff
$$D_{\delta_0} \cup D_{\text{pre}} \models \Gamma_i \models \exists n(\Gamma_i) \land \Gamma_i.$$
tests in the world requires a complex sequencing of world-altering and sensing actions, whose selection and ordering is conditioned upon the outcome of previous sensing actions. We proposed specifying such complex tests in the logic programming language Golag. We then demonstrated that regression could be used both to verify the desired objective of such complex tests, and to generate tests as conditional plans under certain restrictions.

Sensing is integral to the operation of most autonomous agents. The notion of complex and simple tests introduced here extends the body of theoretical work on sensing in dynamical systems, and has practical relevance for building agents for diagnostic problem solving, plan understanding, or simply for mobile cognitive agents that need to interact in complex environments with limited sensing.

Acknowledgments

We thank Eyal Amir for useful comments on an earlier version of this paper. Additionally, we thank Yves Lespérance for useful discussions related to this paper. This research was supported in part by NSF grant NSF 9819116, DARPA grant N66001-97-C-8554-F00004 and by NASA grant NAG2-1337.

References


Weld, D. 1999. Recent advances in AI planning. AI Magazine.