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.

Such a sequence of actions constitutes a *test*. If steam is observed, then the agent knows that there is power at the outlet; however if steam is not observed, the agent may or may not know that there is no power at the electrical outlet. The knowledge the agent acquires from the test will depend on whether the agent knows that the kettle is functioning. Thus, this particular test is only *guaranteed* to provide knowledge about the existence of power at the electrical outlet under one test outcome.

While researchers have extended theories of action to include the notion of *sensing* or *knowledge-producing actions* (e.g., (Scherl & Levesque 1993; Baral & Tran 1998; Golden & Weld 1996; Funge 1998)) and have characterized the effect of sensing actions on an agent's state of knowledge, and even how to plan (e.g., (Stone 1998; Golden & Weld 1996)) and to project (e.g., (De Giacomo & Levesque 1999b)) in certain cases, with sensing actions, they have not addressed the problem of how to reason in a partially observable environment[1]. More generally, they have not examined the problem of how sensing actions can be coupled with knowledge of the relationship between objects in the world to gain further knowledge, and how both sensing actions, and world-altering actions change an agents state of knowledge in the presence of such world knowledge. Further, they have not examined the problem of how to select sensing actions to acquire knowledge of some property of the world that is not directly observable. Perhaps the closest research is that of (Shanahan 1996b; 1996a) who investigates the assimilation of sensing results for a mobile robot in a framework based on the event calculus, (McIlraith 1997) who assimilates observations into situation calculus device models to perform dynamical diagnosis, or (Baral, McIlraith, & Tran 2000) who do likewise in the language $\mathcal{L}$.

In this paper, we examine these issues in a dialect of the situation calculus that has been extended with knowledge-producing actions[2] (Scherl & Levesque 1993), but which does not include state constraints. Following (McIlraith 2000), we add state constraints to this language in order to

[1]Partially-Observable Markov Decision Processes (POMDPs) address this class of problems within a different formalism, but they do not address the testing issues we examine here.

[2]Henceforth referred to simply as *sensing* actions.

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, $Poss(a, s)$ is a distinguished fluent expressing that action a is possible to perform in situation s. A situation calculus theory $\mathcal{D}$ comprises the following sets of axioms:

- foundational axioms of the situation calculus, Σ,
- successor state axioms, $\mathcal{D}_{SS}$,
- action precondition axioms, $\mathcal{D}_{ap}$,
- axioms describing the initial situation, $\mathcal{D}_{S_0}$,
- unique names for actions, $\mathcal{D}_{una}$,
- domain closure axioms for actions, $\mathcal{D}_{dca}$.

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[3]:

$$Poss(a, s) \wedge \gamma_F^+(\vec{x}, a, s) \supset F(\vec{x}, do(a, s)) \quad (1)$$

$$Poss(a, s) \wedge \gamma_F^-(\vec{x}, a, s) \supset \neg F(\vec{x}, do(a, s)), \quad (2)$$

and ramification (state) constraints of the form:

$$\upsilon_F^+(\vec{x}, s) \supset F(\vec{x}, s) \quad (3)$$

$$\upsilon_F^-(\vec{x}, s) \supset \neg F(\vec{x}, s), \quad (4)$$

into **Intermediate Successor State Axioms** of the form:

$$Poss(a, s) \supset [F_i(\vec{x}, do(a, s)) \equiv \Phi_{F_i}^*] \quad \text{where,} \quad (5)$$

$$\begin{aligned}\Phi_{F_i}^* \equiv\ & \gamma_{F_i}^+(\vec{x}, a, s) \vee \upsilon_{F_i}^+(\vec{x}, do(a, s)) \\ & \vee (F(\vec{x}, s) \\ & \quad \wedge\ \neg(\gamma_{F_i}^-(\vec{x}, a, s) \vee \upsilon_{F_i}^-(\vec{x}, do(a, s)))),\end{aligned} \quad (6)$$

I.e., if an action is possible is 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 $\Phi_{F_i}^*$, $\mathcal{R}^*[\Phi_{F_i}^*] = \Phi_{F_i}$, 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 begin turned ($turning\_ign$), 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 produce successor state axioms. For notational convenience we abbreviate: transmission - $trans$, interlock - $intrlk$, solenoid - $solnd$, engine - $engn$, battery - $batt$; ignition system - ign_sys, start system - $strt\_sys$.

$$empty(gas\_tank, s) \supset \neg startable(s) \quad (7)$$

$$ab(intrlk, s) \supset \neg startable(s) \quad (8)$$

$$ab(batt, s) \supset \neg startable(s) \quad (9)$$

$$ab(solnd, s) \supset \neg startable(s) \quad (10)$$

$$ab(starter, s) \supset \neg startable(s) \quad (11)$$

$$auto(trans) \wedge \neg ingear(trans, s) \supset ab(intrlk, s) \quad (12)$$

$$manual(trans) \wedge \neg depressed(clutch, s) \supset ab(intrlk, s) \quad (13)$$

$$turning\_ign(s) \wedge ab(batt, s) \supset \neg noise(engn, s) \quad (14)$$

$$turning\_ign(s) \wedge empty(gas\_tank, s) \supset \neg noise(engn, s) \quad (15)$$

[3] Notational convention: all formulae are universally quantified with maximum scope unless otherwise noted.

$$turning\_ign(s) \wedge \neg ab(solnd, s) \supset noise(solnd, s) \quad (16)$$

$$ab(batt, s) \wedge on(radio, s) \supset \neg noise(radio, s) \quad (17)$$

$$ab(radio, s) \supset \neg noise(radio, s) \quad (18)$$

$$\neg ab(batt, s) \wedge on(lights, s) \supset emits(light, s) \quad (19)$$

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):

$$\begin{aligned} &Poss(a, s) \supset [startable(do(a, s)) \equiv \\ &\quad \neg empty(gas\_tank, do(a, s)) \wedge \neg ab(intrlk, do(a, s)) \\ &\quad \wedge \neg ab(batt, do(a, s)) \wedge \neg ab(solnd, do(a, s)) \\ &\quad \wedge \neg ab(starter, do(a, s))] \end{aligned} \quad (20)$$

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 $empty(gas\_tank, do(a, s))$, $startable(do(a, s))$, and $noise(radio, do(a, s))$.

The notation **Knows**(ϕ, s) (read as ϕ is known in situation s), where ϕ arbitrary formula, is an abbreviation for a formula that uses K. For example **Knows**$(on(block_1, block_2), s)$ abbreviates:

$$\forall s'\ K(s', s) \supset on(block_1, block_2, s').$$

The notation **Kwhether**(ϕ, s) is an abbreviation for a formula indicating that the truth value of ϕ is known.

$$\textbf{Kwhether}(\phi, s) \stackrel{\text{def}}{=} \textbf{Knows}(\phi, s) \vee \textbf{Knows}(\neg\phi, s),$$

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

$$SF(a, s) \equiv \psi(s), \quad (21)$$

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

For the sense action $check\_fuel$ the sensed fluent axiom is:

$$SF(check\_fuel, s) \equiv empty(gas\_tank, s) \quad (22)$$

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

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

Successor State Axiom for K

$$\begin{aligned} &Poss(a, s) \supset [K(s'', do(a, s)) \equiv \\ &\quad \exists s'.\ Poss(a, s') \wedge K(s', s) \wedge (s'' = do(a, s')) \wedge \\ &\qquad [SF(a, s') \equiv SF(a, s)] \end{aligned} \quad (23)$$

which says that after doing action a in situation s, the agent thinks it could be in a situation s'' iff $s'' = do(a, s')$ 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 $check\_radio\_noise$, $Poss(check\_radio\_noise, s) \equiv inside(car, s)$, 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.

$(on(radio, s), check\_radio\_noise)$ 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 HYP. We restrict the hypotheses, $H(s) \in HYP$ 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 $\{ab(batt, s), ab(solnd, s), empty(gas\_tank, s)\}$.

Definition 2 (Test for Hypothesis Space HYP)
A test (I, a) is a test for hypothesis space HYP in situation s iff $\mathcal{D} \wedge I \wedge Poss(a, s) \wedge H(s)$ is satisfiable for every $H(s) \in HYP$.

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 $\mathcal{D}$ entails the safety constraint $\neg explosion(s)$ and the axiom $sparks(s) \wedge gas\_leak(s) \supset explosion(s)$, and that our hypothesis space is $\{gas\_leak(s), ab(spark\_plug, s)\}$. A reasonable test for $ab(spark\_plug, s)$ 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 α of the test (I, a) **confirms** *$H(s) \in HYP$ iff $\mathcal{D} \wedge I \wedge Poss(a, s) \wedge H(s)$ is satisfiable and $\mathcal{D} \wedge I \wedge Poss(a, s) \models$* **Knows**$(H \supset \alpha, s)$. *$\alpha$* **refutes** *$H(s)$ iff $\mathcal{D} \wedge I \wedge Poss(a, s) \wedge H(s)$ is satisfiable and $\mathcal{D} \wedge I \wedge Poss(a, s) \models$* **Knows**$(H \supset \neg\alpha, s)$.

If the outcome of test $(on(radio, s), check\_radio\_noise)$ is $noise(radio, do(a, s))$, then our test refutes the hypothesis $ab(batt, s)$, following Axiom (17), and we can eliminate $ab(batt, s)$ from our hypothesis space, HYP.

Observe that a test outcome that refutes an hypothesis $H(s)$ allows us to eliminate it from HYP. 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 $\neg noise(radio, do(a, s))$, then by the definition, this would have confirmed the hypothesis $ab(batt, s)$, 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 HYP, i.e., which will refute at least one hypothesis in HYP, regardless of the test outcome.

Definition 4 (Discriminating Tests)
A test (I, a) is a discriminating test for the hypothesis space HYP iff $\mathcal{D} \wedge I \wedge Poss(a, s) \wedge H(s)$ is satisfiable for all $H(s) \in HYP$, and there exists $H_i(s)$, $H_j(s) \in HYP$ such that the outcome α of test (I, a) refutes either $H_i(s)$ or $H_j(s)$, no matter what that outcome might be.

Proposition 2
After we perform a discriminating test, (I, a), **Knows**$(\neg H_j, s)$*, for some $H_j(s) \in HYP$.*

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 HYP is minimal iff for no proper subconjunct I' of I is (I', a) a discriminating test for HYP.

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 $H_i(s)$ and $\neg H_i(s) \in HYP$ iff $\mathcal{D} \wedge I \wedge Poss(a, s) \wedge H(s)$ is satisfiable for all $H(s) \in HYP$ and the outcome α of test (I, a) refutes either $H_i(s)$ or $\neg H_i(s)$, no matter what that outcome might be.

Proposition 3
After we perform an individual discriminating test (I, a), **Kwhether**(H_i, s) *for some $H_i \in HYP$.*

The test $(\{\}, check\_fuel)$ is such a test. The outcome will be one of $\neg empty(gas\_tank, do(a, s))$ or $empty(gas\_tank, do(a, s))$. Thus, as the result of performing $check\_fuel$ in the physical world, the agent **Kwhether**$(empty(gas\_tank, s))$.

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 HYP, but which cannot be guaranteed to do so. Given a particular outcome α, a relevant test may refute a subset of the hypotheses in the hypothesis space HYP, 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. $(on(radio, s), check_radio_noise)$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 $\mathcal{D} \wedge I \wedge Poss(a, s) \wedge H(s)$ is satisfiable for all $H(s) in HYP$, and the outcome α 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 $\mathcal{D}$ and outcome α of the test (I, a) is any $H(s) \in HYP$ such that $\mathcal{D} \wedge I \wedge Poss(a, s) \wedge H(s) \wedge \alpha$ is satisfiable.

Proposition 4 (Eliminating C-B Hypotheses)
The outcome α of a test (I, a) eliminates those consistency-based hypotheses, $H(s) \in HYP$ that are refuted by test outcome α.

Definition 9 (Abductive Hypothesis Space)
An abductive hypothesis for $\mathcal{D}$ and outcome α of the test (I, a) is any $H(s) \in HYP$ such that $\mathcal{D} \wedge I \wedge Poss(a, s) \wedge H(s)$ is satisfiable, and $\mathcal{D} \wedge I \wedge Poss(a, s) \wedge H(s) \models \alpha$.

Proposition 5 (Eliminating Abductive Hypotheses)
The outcome α of a test (I, a) eliminates those abductive hypotheses, $H(s) \in HYP$ that are not confirmed by test outcome α.

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 α 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 (alGOl 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 δ is a complex action expression. $Do(\delta, s, s')$ is intended to mean that the agent's doing action δ in situation s leads to a (not necessarily unique) situation s'. The inductive definition of Do includes the following cases:

$Do(a, s, s')$ — simple actions
$Do(\phi?, s, s')$ — tests (referred to as G-tests in this paper)
$Do([\delta_1; \delta_2], s, s')$ — sequences
$Do([\delta_1 | \delta_2], s, s')$ — nondeterministic choice of actions
$Do((\Pi x)\delta, s, s')$ — nondeterministic choice of parameters
$Do(\textbf{if } \phi \textbf{ then } \delta_1 \textbf{ else } \delta_2, s, s')$– conditionals, where we restrict ϕ to a G-test
$Do(\textbf{while } \phi \textbf{ do } \delta, s, s')$ — 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[4].

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 $\{ab(intrlk, s), empty(gas\_tank, s), ab(batt, s), ab(solnd, s), ab(ign\_wires, s), ab(starter, s)\}$. In a diagnostic application such as this one, Golog procedures may also be written to combine testing with repair.

proc CARWONTSTART
  if (¬ startable) **then** CHECKINTERLOCK;

[4]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(INTRLK)) **then** CHECK_GAS_TANK;
if (¬ EMPTY(GAS_TANK)) **then** CHECKBATTERY;
if (¬ AB(BATT)) **then** CHECKSOLENOID;
if (¬ AB(SOLND)) **then** CHECKIGNWIRES;
if (¬ AB(IGN_WIRES)) **then** CHECKSTARTER;
if (¬ AB(STARTER)) **then** CHECKENGINE
end if end if end if end if end if end if end if
endProc

proc CHECKBATTERY
TURN_ON(RADIO); CHECK_RADIO_NOISE;
if (¬ NOISE(RADIO))
then TURN_ON(LIGHTS); CHECK_LIGHTS
end if
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 them 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 (Lespérance 1994) and (Reiter 2000). Other 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, $\mathbf{Kwhether}(H, s)$ holds, where s is the situation resulting from the execution of the Golog procedure, i.e. $Do(\text{CARWONTSTART}, S_0, s)$. Thus, we would like to be able to entail $\bigvee_{H \in HYP} \mathbf{Kwhether}(H, s)$, and in particular $\mathbf{Kwhether}(empty(gas\_tank), s)$, for example. A verification that the procedure is a discriminating test would involve ensuring that for at least one H, $\mathbf{Knows}(\neg H, s)$ holds in the final situation, i.e., $\bigvee_{H \in HYP} \mathbf{Knows}(\neg H, s)$.

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 (i.e. a term built on S_0 with the function *do* and ground action terms) s_{gr}, the problem is to determine whether the axiomatization of the domain $\mathcal{D}$ entails $G(s_{gr})$ 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 $G(s_{gr})$, i.e.,$\mathcal{R}(G(s_{gr}))$. 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, $\mathcal{D}_{S_0}$. In our case G will be a formula made up of subformulae of the form $\mathbf{Kwhether}(H, s)$ or $\mathbf{Knows}(\neg H, s)$, where H is an hypothesis.

The regression operator $\mathcal{R}$ is defined relative to a set of successor state axioms $\mathcal{D}_{ss}$. The first four parts of the definition of the regression operator[5], $\mathcal{R}$ concern world-altering actions and are taken from (Reiter 2000).

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

ii. When F is a relational fluent (other than K) atom whose successor state axiom in $\mathcal{D}_{SS}$ is

$$Poss(a, s) \supset [F(x_1, \ldots, x_n, do(a, s)) \equiv \Phi_F]$$

then

$$\mathcal{R}[F(t_1, \ldots, t_n, do(\delta, \sigma))] = \Phi_F|^{x_1, \ldots, x_n, a, s}_{t_1, \ldots, t_n, \delta, \sigma}$$

iii. Whenever W is a formula,
$\mathcal{R}[\neg W] = \neg\mathcal{R}[W]$,
$\mathcal{R}[(\forall v)W] = (\forall v)\mathcal{R}[W]$,
$\mathcal{R}[(\exists v)W_1] = (\exists v)\mathcal{R}[W_1]$.

iv. Whenever W_1 and W_2 are formulas,
$\mathcal{R}[W_1 \wedge W_2] = \mathcal{R}[W_1] \wedge \mathcal{R}[W_1]$,
$\mathcal{R}[W_1 \vee W_2] = \mathcal{R}[W_1] \vee \mathcal{R}[W_1]$,
$\mathcal{R}[W_1 \supset W_2] = \mathcal{R}[W_1] \supset \mathcal{R}[W_1]$.

Following (Scherl & Levesque 1993), additional steps are needed to extend the regression operator to sensing actions[6]. Two definitions are needed for the specification to follow. When φ 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 φ and inserting s into that argument

[5]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 *regressable*. This concept is fully defined in (Reiter 2000).

[6]Regression of sensing actions that make known the denotation of a term (e.g. an action of reading a number on a paper) is not discussed here.

position. The reverse operation φ^{-1} is the result of removing the last argument position from all the fluents in φ.

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 definitions below, s' is a new situation variable.

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

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

An additional operator $\mathcal{C}$ 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, sequencing, and conditionals. We also add the empty action **noOp** or [] (names for the same operation). Also note that $\pi_a(\vec{x}, s)$ stands for the preconditions of $a(\vec{x})$ as specified in the action precondition axiom, $\mathcal{D}_{ap}$, $Poss(a(\vec{s}), s) \equiv \pi_a(\vec{x}, s)$.

viii. $\mathcal{C}(\mathbf{noOp}, W, s) = W(s)$

ix. $\mathcal{C}([a(\vec{x}); \delta], W, s) = \pi_a(\vec{x}, s) \wedge \mathcal{C}(\delta, W, do(a(\vec{x}), s))$ where $a(\vec{x})$ is a ground non-sensing simple action term.

x. $\mathcal{C}([\mathbf{if}\ \phi\ (\vec{x})\ \mathbf{then}\ \delta_1\ \mathbf{else}\ \delta_2], W, s) =$
$\mathbf{Kwhether}(\phi(\vec{x}), s) \wedge$
$[\mathbf{Knows}(\phi(\vec{x}), s) \supset \mathcal{C}(\delta_1, W, s)] \wedge$
$[\mathbf{Knows}(\neg\phi(\vec{x}), s) \supset \mathcal{C}(\delta_2, W, s)]$

We are assuming that the agent is able[8] to execute the Golog test procedure. In particular, the programmer (of the test procedure) must have ensured that at the point where an $[\mathbf{if}\ \phi\ (\vec{x})\ \mathbf{then}\ \delta_1\ \mathbf{else}\ \delta_2]$ statement is encountered, the executing agent must $\mathbf{Kwhether}(\phi, s)$. If not, the procedure will fail.

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

Theorem 2 *For any Golog procedure δ, consisting of simple actions, sequences, and conditionals, and G an arbitrary closed regressable formula that may include knowledge operators:*

$$\mathcal{D} \models \exists s(Do(\delta, S_0, s) \wedge G(s)) \text{ iff}$$
$$\mathcal{D}_{S_0} \cup \mathcal{D}_{una} \models \mathcal{R}^*(\mathcal{C}(\delta, G, S_0))$$

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 ($\mathcal{D}_{ss}$) are only used in the regression procedure. This theorem can be extended to likewise verify other properties of our Golog procedures.

[7]The $\mathcal{C}$ operator introduced here is based on (but generalizes) the E operator of (Lespérance 1994).

[8]See (Lespérance *et al.* 2000) for a discussion of ability and Golog programs. Related issues are discussed in (Lespérance 1994; Lakemeyer 1999).

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 ϕ in $[\mathbf{if}\ \phi\ (\vec{x})\ \mathbf{then}\ \delta_1\ \mathbf{else}\ \delta_2]$) must be a sensed formula. Thus we can require that prior to any conditional with the G-test ϕ, there must be an action a such that a is a sensing action and $\mathcal{D} \models SF(a,s) \equiv \phi(s)$. This guarantees that the program executing the test will always $\mathbf{Kwhether}(\phi, s)$ 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 Γ_i as:

$$\Gamma_0 \stackrel{\text{def}}{=} G(s)$$

$$\begin{aligned}
\Gamma_i \stackrel{\text{def}}{=}\ & \\
& \exists a([\exists \vec{x}\,(a = A_1(\vec{x}) \wedge \pi_{A_1}(\vec{x}) \vee \ldots \\
& \quad \vee \exists \vec{x}\,(a = A_n(\vec{x}) \wedge \pi_{A_n}(\vec{x}))] \\
& \wedge\ \mathcal{R}(\Gamma_{i-1}(do(a,s)))) \vee \\
& \exists a([\exists \vec{x}\,(a = A_1^s(\vec{x}) \wedge \pi_{A_1^s}(\vec{x}) \wedge (SF(a,s) \equiv \phi_1(s)) \wedge \\
& \mathcal{R}(\phi_1(\vec{x}, do(a,s)) \supset \Gamma_{i-1}(do(a,s))) \wedge \\
& \quad \mathcal{R}(\neg\phi_1(\vec{x}, do(a,s)) \supset \Gamma_{i-1}(do(a,s)))] \\
& \wedge \ldots \wedge \\
& \exists a([\exists \vec{x}\,(a = A_m^s(\vec{x}) \wedge \pi_{A_m^s}(\vec{x}) \wedge (SF(a,s) \equiv \phi_m(s)) \wedge \\
& \mathcal{R}(\phi_m(\vec{x}, do(a,s)) \supset \Gamma_{i-1}(do(a,s))) \wedge \\
& \quad \mathcal{R}(\neg\phi_m(\vec{x}, do(a,s)) \supset \Gamma_{i-1}(do(a,s)))]
\end{aligned}$$

Each Γ_i is true if there is a plan of length i starting in s and leading to a state satisfying G (Reiter 1995; Lespérance 1994). The following theorem (essentially Theorem 3 of (Lespérance 1994)) establishes the soundness and completeness of the regression-based test planning method.

Theorem 3 *For Golog procedure δ in normal form and G, an arbitrary closed regressable formula that may include knowledge operators:*

$$\mathcal{D} \models \exists s(Do(\delta, S_0, s) \wedge G(s)) \text{ iff } \textit{for some } n$$
$$\mathcal{D}_{S_0} \cup \mathcal{D}_{una} \models \Gamma_0(S_0) \vee \ldots \vee \Gamma_n(S_0)$$

This regression-based finite horizon method of generating and evaluating all normal form conditional plans of greater and greater size is certainly not designed for efficiency, but the results can serve as the foundation for building more efficient regression-based complex-test planning methods, much as similar results have served as the foundation for relatively more efficient regression based planning methods (McDermott 1991; Lespérance 1994; Rosenschein 1981). In future work we will evaluate the extension of current state of the art planning techniques based on SAT and Graphplan, to address the planning problems raised in this paper (Weld 1999).

Summary

In this paper we presented results towards a formal theory of testing for dynamical systems, specified in the language of the situation calculus. Our first contribution was to address the ramification problem for sensing actions. We then defined the notion of a test, examining how a test can be designed and how the outcome of different types of tests affect an agent's state of knowledge. The realization of many 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 Golog. 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-P00004 and by NASA grant NAG2-1337.

References

Baral, C., and Tran, S. 1998. Formalizing sensing actions: a transition function based approach. In *Proc. of AAAI 98 Fall Symposium on Cognitive Robotics*.

Baral, C.; McIlraith, S.; and Tran, S. 2000. Formulating diagnostic problem solving using an action language with narratives and sensing. In *Proc. Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR'2000)*. To appear.

De Giacomo, G., and Levesque, H. J. 1999a. An incremental interpreter for high-level programs with sensing. In Levesque, H. J., and Pirri, F., eds., *Logical Foundations for Cognitive Agents: Contributions in Honor of Ray Reiter*. Berlin: Springer. 86–102.

De Giacomo, G., and Levesque, H. J. 1999b. Progression and regression using sensors. In *Proc. Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*, 160–165.

Funge, J. 1998. *Making Them Behave: Cognitive Models for Computer Animation*. Ph.D. Dissertation, Department of Computer Science, University of Toronto.

Golden, K., and Weld, D. 1996. Representing sensing actions: The middle ground revisited. In *Proc. Fifth Intnl. Conf. on Principles of Knowledge Representation and Reasoning (KR'96)*.

Lakemeyer, G. 1999. On sensing and off-line interpreting in GOLOG. In Levesque, H. J., and Pirri, F., eds., *Logical Foundations for Cognitive Agents: Contributions in Honor of Ray Reiter*. Berlin: Springer. 173–189.

Lespérance, Y.; Levesque, H. J.; Lin, F.; and Scherl, R. B. 2000. Ability and knowing how in the situation calculus. *Studia Logica*. To appear.

Lespérance, Y. 1994. An approach to the synthesis of plans with perception acts and conditionals. In *Working Notes of the Canadian Workshop on Distributed Artificial Intelligence*.

Levesque, H.; Reiter, R.; Lespérance, Y.; Lin, F.; and Scherl, R. 1997. GOLOG: A logic programming language for dynamic domains. *The Journal of Logic Programming* 31:59–84.

Levesque, H. 1996. What is planning in the presence of sensing? In *Proc. Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, 1139–1146.

Lin, F., and Reiter, R. 1994. State constraints revisited. *Journal of Logic and Computation* 4(5):655–678. Special Issue on Action and Processes.

Lobo, J. 1998. COPLAS: a COnditional PLAnner with Sensing actions. In *Working Notes of the AAAI98 Fall Symposium on Cognitive Robotics*, 109–116.

Manna, Z., and Waldinger, R. 1987. How to clear a block: A theory of plans. *Journal of Automated Reasoning* 3:343–377.

McDermott, D. 1991. Regression planning. *International Journal of Intelligent Systems* 6:356–416.

McIlraith, S., and Reiter, R. 1992. On tests for hypothetical reasoning. In W. Hamscher, L. C., and de Kleer, J., eds., *Readings in model-based diagnosis*. Morgan Kaufmann. 89–96.

McIlraith, S. 1994. Generating tests using abduction. In *Proc. Fourth International Conference on Principles of Knowledge Representation and Reasoning (KR'94)*, 449–460.

McIlraith, S. 1997. *Towards a Formal Account of Diagnostic Problem Solving*. Ph.D. Dissertation, Department of Computer Science, University of Toronto, Toronto, Ontario, Canada.

McIlraith, S. 2000. A closed-form solution to the ramification problem (sometimes). *Artificial Intelligence* 116(1–2):87–121.

Moore, R. 1985. A formal theory of knowledge and action. In Hobbs, J. B., and Moore, R. C., eds., *Formal Theories of the Commonsense World*. Ablex Publishing Corp. 319–358.

Ramsey, D. 1999. *The Complete Idiot's Guide to Trouble-Free Car Care*. Alpha Books.

Reiter, R. 1991. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of J. McCarthy*.

Reiter, R. 1995. CS2532S course notes. Unpublished manuscript.

Reiter, R. 2000. *Knowledge in Action: Logical Foundations for Describing and Implementing Dynamical Systems*. In preparation. Draft available at http://www.cs.toronto.edu/˜cogrobo/.

Rosenschein, S. 1981. Plan synthesis: A logical perspective. In *Proc. Seventh International Joint Conference on Artificial Intelligence (IJCAI-81)*, 331–337.

Scherl, R., and Levesque, H. 1993. The frame problem and knowledge producing actions. In *Proc. Eleventh National Conference on Artificial Intelligence (AAAI-93)*, 689–695.

Shanahan, M. 1996a. Noise and the common sense informatic situation for a mobile robot. In *Proc. Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, 1098–1103.

Shanahan, M. 1996b. Robotics and the common sense informatic situation. In *Proc. European Conference on Artificial Intelligence (ECAI-96)*, 684–688.

Stone, M. 1998. Abductive planning with sensing. In *Proc. Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, 631–636.

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