Table of Contents I

Diagnostic Agents
- Recording the History of a Domain
- Defining Explanations
- Computing Explanations

Reading

- Read Chapter 10, *Diagnostic Agents*, in the KRR book.

Diagnostic Agents

- Goal: build agents capable of finding explanations to unexpected observations.
- To do this, we need:
  - a model of what is expected in the first place,
  - a method of making and recording observations,
  - and a method of detecting when reality doesn't match expectations.

Two Types of Actions

- Previously, we were only interested in **agent actions**.
- Now we are also interested in modeling **exogenous actions**, which are those performed by nature or by other agents.
- Therefore, we will split our actions into these two types:

```
sort
  #action = #agent_action + #exogenous_action.
```

Simplifying Assumptions

1. The agent is capable of making correct observations, performing actions, and recording these observations and actions.
2. *Normally* the agent is capable of observing all relevant exogenous actions occurring in its environment.

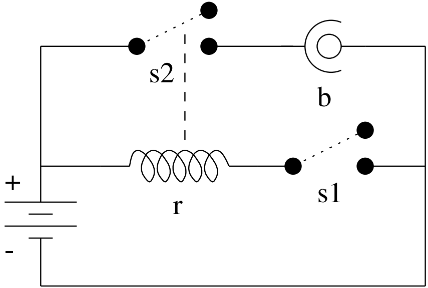
Note that the second assumption is defeasible.

The Diagnostic Problem

- A **symptom** consists of a recorded history of the system such that its last collection of observations is unexpected; i.e., it contradicts the agent's expectations.
- An **explanation** of a symptom is a collection of unobserved past occurrences of exogenous actions which may account for the unexpected observations.
- **Diagnostic Problem**: Given a description of a dynamic system and a symptom, find a possible explanation of the latter.

Example of a Diagnostic Problem

Consider an agent controlling a simple electrical system:



It is aware of two exogenous actions: *break* (breaks bulb) and *surge* (breaks relay and breaks bulb if bulb unprotected).

What Is Our Intuition?

Suppose initially:

- the bulb is protected
- the bulb is OK
- the relay is OK
- agent closes s_1

Agent expects the that the relay would become active causing s_2 to close and the bulb to emit light. What should it think if it observes that the light is not lit?

Possible explanations:

1. *break* occurred.
2. *surge* occurred.
3. *break* and *surge* occurred in parallel.

Humans tend to prefer minimal explanations.

- If the agent observes that the bulb is OK, then the only possible minimal explanation is *surge*.
- If the bulb was observed to be broken, then *break* is the explanation.
- If the bulb had not been protected, then both explanations would be valid.

Recording History

- In order to reason about the past, the agent must have a record of the actions and observations it made.
- This recorded history defines a collection of paths that can be viewed as the system's possible pasts.
- Complete knowledge = 1 path

Recorded History — Syntax

(This is the way we record observations and actions.)

The **recorded history** Γ_{n-1} of a system up to a current step n is a collection of **observations** that come in one of the following forms:

1. $obs(f, true, i)$ — fluent f was observed to be true at step i; or
2. $obs(f, false, i)$ — fluent f was observed to be false at step i; or
3. $hpd(a, i)$ — action a was performed by the agent or observed to happen at step i

where i is an integer from the interval $[0, n)$.

Recorded History — Semantics

(This tells us how to match the set of *obs* and *hpd* statements with a transition diagram.)

A path $\langle \sigma_0, a_0, \sigma_1, \ldots, a_{n-1}, \sigma_n \rangle$ in the transition diagram $\mathcal{T}(\mathcal{SD})$ is a **model of a recorded history** Γ_{n-1} of dynamic system $\mathcal{SD}$ if for any $0 \leq i < n$

1. $a_i = \{a : hpd(a, i) \in \Gamma_{n-1}\}$;
2. if $obs(f, true, i) \in \Gamma_{n-1}$ then $f \in \sigma_i$;
3. if $obs(f, false, i) \in \Gamma_{n-1}$ then $\neg f \in \sigma_i$.

We say that Γ_{n-1} is **consistent** if it has a model.

Entailment

(This tells us when a recorded history entails a fluent literal.)

$M \models h(l, i)$
: A fluent literal l **holds** in a model M of Γ_{n-1} at step $i \leq n$ if $l \in \sigma_i$;

$\Gamma_{n-1} \models h(l, i)$
: Γ_{n-1} **entails** $h(l, i)$ if, for every model M of Γ_{n-1}, $M \models h(l, i)$.

Example: Briefcase Domain

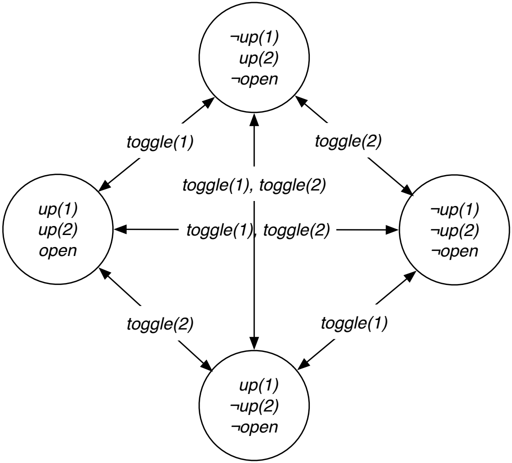
$$
\begin{array}{l}
toggle(C) \;\; \textbf{causes} \;\; up(C) \;\; \textbf{if} \;\; \neg up(C) \\
toggle(C) \;\; \textbf{causes} \;\; \neg up(C) \;\; \textbf{if} \;\; up(C) \\
open \;\; \textbf{if} \;\; up(1), up(2)
\end{array}
$$

Suppose that, initially, clasp 1 was fastened and the agent unfastened it. The corresponding recorded history is:

$$
\Gamma_0 \begin{cases} obs(up(1), false, 0). \\ hpd(toggle(1), 0). \end{cases}
$$

What are the possible models of Γ_0 that satisfy this history?

Transition Diagram for Briefcase Domain



Γ_0 Has Two Models

$$M_1 = \langle \{\neg up(1), \neg up(2), \neg open\}, toggle(1), \{up(1), \neg up(2), \neg open\} \rangle$$

$$M_2 = \langle \{\neg up(1), up(2), \neg open\}, toggle(1), \{up(1), up(2), open\} \rangle$$

Although we have a consistent history, our knowledge is incomplete.
However, we can conclude that clasp 1 is up at step 1 because

$$\Gamma_0 \models holds(up(1), 1)$$

An Inconsistent History

$$\Gamma_0 \begin{cases} obs(up(1), true, 0) \\ obs(up(2), true, 0) \\ hpd(toggle(1), 0) \\ obs(open, true, 1) \end{cases}$$

There is no path in our diagram that we can follow in this situation.

System Configuration

- An agent just performed its n^{th} action.
- The recorded history is Γ_{n-1}
- The agent observes the values of fluents at step n; we'll call these observations O^n.
- The pair $\mathcal{C} = \langle \Gamma_{n-1}, O^n \rangle$ is often referred to as the **system configuration**.

Agent Loop

- If the new observations are consistent with the agent's view of the world (i.e., $\mathcal{C}$ is consistent), then the observations simply become part of the recorded history.
- Otherwise, it seeks an explanation which is that some exogenous action occurred that the agent did not observe.

Possible Explanation

- A configuration $\mathcal{C} = \langle \Gamma_{n-1}, O^n \rangle$ is called a **symptom** if it is inconsistent, i.e. has no model.
- A **possible explanation** of a symptom $\mathcal{C}$ is a set $\mathcal{E}$ of statements $occurs(a, k)$ where a is an exogenous action, $0 \leq k < n$, and $\mathcal{C} \cup \mathcal{E}$ is consistent.

Example: Diagnosing the Circuit I

Signature, written in SPARC format:

```
#step = 0..n.
#boolean = {true, false}.

% Components
#bulb = {b}.
#relay = {r}.
#comp = #bulb + #relay.
#agent_switch = {s1}.
#switch = [s][1..2].
```

Example: Diagnosing the Circuit II

```
% Fluents
#inertial_fluent = prot(#bulb) +
                   closed(#switch) +
                   ab(#comp).
#defined_fluent = active(#relay) +
                  on(#bulb).
#fluent = #inertial_fluent + #defined_fluent.

%Actions
#agent_action = close(#agent_switch).
#exogenous_action = {break, surge}.
#action = #agent_action + #exogenous_action.
```

System Description

- Normal Function

$$\begin{aligned}
&close(s_1) \textbf{ causes } closed(s_1)\\
&active(r) \textbf{ if } closed(s_1), \neg ab(r)\\
&closed(s_2) \textbf{ if } active(r)\\
&on(b) \textbf{ if } closed(s_2), \neg ab(b)\\
&\textbf{impossible } close(s_1) \textbf{ if } closed(s_1)
\end{aligned}$$

- Malfunction

$$\begin{aligned}
&break \textbf{ causes } ab(b)\\
&surge \textbf{ causes } ab(r)\\
&surge \textbf{ causes } ab(b) \textbf{ if } \neg prot(b)
\end{aligned}$$

A History

$$\Gamma_0 \begin{cases} obs(closed(s_1), false, 0) \\ obs(closed(s_2), false, 0) \\ obs(ab(b), false, 0) \\ obs(ab(r), false, 0) \\ obs(prot(b), true, 0) \\ hpd(close(s_1), 0) \end{cases}$$

- What is the model of this history?
- What does it entail about the bulb?
- Let's look at the program:
  `http://pages.suddenlink.net/ykahl/s_circuit.txt`

Example: Symptom and Explanations

- Suppose that the agent observes that the bulb is not lit.
- This means that

$$\mathcal{C} = \langle \Gamma_0, obs(on(b), false, 1) \rangle$$

  is a symptom.
- This symptom may have three possible explanations:

$$\begin{aligned} \mathcal{E}_1 &= \{occurs(surge, 0)\}, \\ \mathcal{E}_2 &= \{occurs(break, 0)\}, \\ \mathcal{E}_3 &= \{occurs(surge, 0), occurs(break, 0)\}. \end{aligned}$$

- Actions *break* and *surge* are the only exogenous actions available in our language, and $\mathcal{E}_1, \mathcal{E}_2$, and $\mathcal{E}_3$ are the only sets such that $\mathcal{C} \cup \mathcal{E}_i$ is consistent.

Computing Explanations

To compute explanations, our program must be able to

1. Recognize that there is a symptom.
2. Consider possible, unobserved exogenous actions as explanations.

all_clear($\mathcal{SD}, \mathcal{C}$): Detecting a Symptom

To detect a symptom, we add the following axioms to our system description and configuration:

```
%% Full Awareness Axiom:
holds(F,0) | -holds(F,0) :- #inertial_fluent(F).

%% Take what actually happened into account:
occurs(A,I) :- hpd(A,I).

%% Reality Check:
:- obs(F,true,I), -holds(F,I).
:- obs(F,false,I), holds(F,I).
```

with I ranging over $[0, n]$. If the new program is consistent, then all's well. Otherwise, diagnostics are required.

diagnose($\mathcal{SD}, \mathcal{C}$): Finding Explanations

To create a program which creates explanations, we take program **all_clear**($\mathcal{SD}, \mathcal{C}$) and add the following rules:

```
%% The generator:
occurs(A,K) | -occurs(A,K) :- #exogenous_action(A),
                              K < n.

%% This rule isolates actions that may be
%% part of an explanation:
expl(A,I) :- #exogenous_action(A),
             occurs(A,I),  % Action A might have occurred
             not hpd(A,I). % Action A was not observed
```

Better Explanations

As with minimal plans, minimal explanations can be found by replacing the disjunctive generation rule with a cr-rule:

```
occurs(A,K) :+ #exogenous_action(A),
                K < n.
```

or the minimize statement of Clingo:

$$minimize\{occurs(A, S) : action(exogenous, A) : step(S)\}.$$

Better Explanations: Beyond Cardinality

- Suppose we had another action, make_coffee, in our program which had nothing to do with the proper functioning of the circuit.
- If we wish to eliminate such irrelevant actions, but not necessarily all non-minimal explanations, we can impart our agent with some concept of relevance

```
relevant(break,on(b)).
relevant(surge,on(b)).
% Note we do not have
% relevant(make_coffee,on(b)).
```

and add the following constraint:

```
:- #exogenous_action(X),
   occurs(X,I),
   not hpd(X,I),
   not relevant(X,on(b)).
```