10
Diagnostic Agents

In this chapter we discuss how to build agents capable of finding explanations of unexpected observations. To do that we divide actions of our domain into two disjoint classes: **agent actions** and **exogenous actions**. As expected the former are those performed by the agent associated with the domain, and the latter are those performed by nature or by other agents.[1] As usual we make two 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 – some exogenous actions can remain unobserved. These assumptions hold in many realistic domains and are suitable for a broad class of applications. In other domains, however, the effects of actions and the truth-values of observations can only be known with a substantial degree of uncertainty, which cannot be ignored in the modeling process. We comment on such situations in Chapter 11, which deals with probabilistic reasoning.

In our setting a typical **diagnostic problem** is informally specified as follows:

- 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 occurences of exogenous actions that may account for the unexpected observations.

[1] If our agent is part of a multi-agent system, it is convenient to view actions of other agents as exogenous, just as we would view the occurrence of any other naturally occurring action.

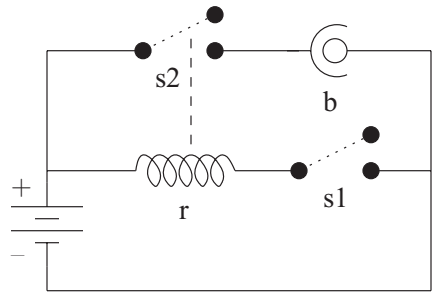

Figure 10.1. Circuit $\mathcal{AC}$

This notion of explanation is closely connected with our second simplifying assumption. Recall that, although assumptions about the agent's ability to perform and observe, as well as its knowledge of causal laws, are not defeasible, the completeness of its observations of exogenous actions is defeasible. Accordingly, the agent realizes that it may miss some of the exogenous actions and attributes its wrong prediction to such an omission. A collection of missing occurrences of exogenous actions that may account for the discrepancy is therefore viewed as an explanation.

- **Diagnostic Problem:** Given a description of a dynamic system and a symptom, find a possible explanation of the latter.

We illustrate this intuition by the following example.

Example 10.0.1. *(A Diagnostic Problem)*
Consider an agent controlling a simple electrical system consisting of circuit $\mathcal{AC}$ (see Fig. 10.1). We assume that switches s_1 and s_2 are mechanical components that cannot become damaged. Relay r is a magnetic coil. If not damaged, it is activated when s_1 is closed, causing s_2 to close. An undamaged bulb b emits light if s_2 is closed. The agent is aware of two exogenous actions relevant to its work: $break$, which causes the circuit bulb to become faulty, and $surge$, which damages the relay and also the bulb if the latter is not protected.

Suppose that at the beginning of the agent's activity the system is in the state depicted by Figure 10.1: The bulb is protected, both the bulb and the relay are known to be OK, and the agent closes switch s_1. The agent expects that this action will activate relay r and that, as a consequence, switch s_2 will become closed and bulb b will emit light. Assume now that the agent is surprised by an unexpected observation – the light is not lit. There are

three natural explanations for this phenomenon: The bulb broke, the relay broke, or both broke. In other words one or both of the actions $break$ and $surge$ occurred in parallel with the agent's closing the switch. It seems that the commonsense reasoner usually ignores the third explanation and only concentrates on the first two. This is probably due to some notion of a minimal or best explanation that usually depends on ordering explanations as determined by the domain. In our case the first two explanations are minimal with respect to set-theoretic and cardinality-based orderings of explanations.

If, in addition to the initial observations, the agent was to observe that the bulb was OK, then the only possible minimal explanation of our unexpected observation would be the occurrence of $surge$. If the bulb was observed to be broken, then the only possible minimal explanation would be the occurrence of $break$. Of course, if initially the bulb was not protected, then both minimal explanations would still be valid. To find out the correct one, the agent would need to perform more "testing" actions (e.g., replace the bulb, etc).

In what follows we make this intuition precise. We start by presenting the syntax and semantics of the system's recorded history and show how it can be used to predict the current state of the system and to discover and explain unexpected observations.

10.1 Recording the History of a Domain

As mentioned earlier, we assumed that, in addition to a description of the transition diagram that represents possible trajectories of the system, the knowledge base of a diagnostic agent contains the system's **recorded history** – observations made by the agent together with a record of its own actions.

The recorded history defines a collection of paths in the diagram that, from the standpoint of the agent, can be interpreted as the system's possible pasts. If the agent's knowledge is complete (i.e., contains complete information about the initial state and the occurrences of actions) and the system's actions are deterministic, then there is only one such path. The following definitions describe the syntax and semantics of the recorded history of a dynamic system $\mathcal{SD}$ up to the current step n.

Definition 10.1.1. *(Recorded History – Syntax)*
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* $[0, n)$.

Definition 10.1.2. *(Recorded History – Semantics)*
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.*

Definition 10.1.3. *(Entailment)*

- *A fluent literal* l **holds** *in a model* M *of* Γ_{n-1} *at step* $i \leq n$ *(denoted by* $M \models h(l, i)$*) if* $l \in \sigma_i$;
- Γ_{n-1} **entails** $h(l, i)$ *(denoted by* $\Gamma_{n-1} \models h(l, i)$*) if, for every model* M *of* Γ_{n-1}, $M \models h(l, i)$.

Example 10.1.1. *(Recorded History in the Briefcase Domain)*
Let's look back at our briefcase example from Section 8.5.1. The system description, $\mathcal{D}_{bc}$, contains the agent's knowledge about the 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

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

There are two models of Γ_0 that satisfy this history. In both models, our action, a_0, is $toggle(1)$, but our states, σ_0 and σ_1, are different. Path $\langle \sigma_0, toggle(1), \sigma_1 \rangle$ is a model of Γ_0 if

$$M_1 \begin{cases} \sigma_0 = \{\neg up(1), \neg up(2), \neg open\} \\ \sigma_1 = \{up(1), \neg up(2), \neg open\} \end{cases}$$

or

$$M_2 \begin{cases} \sigma_0 = \{\neg up(1), up(2), \neg open\} \\ \sigma_1 = \{up(1), up(2), open\} \end{cases}$$

Since Γ_0 has models, it is consistent with respect to the transition diagram for the briefcase domain in Figure 8.8. Although we have a consistent history, our knowledge is not complete. There are two possible trajectories consistent with the agent's recorded history Γ_0. At the end of one trajectory, the briefcase is open; at the end of the other, it is not. An intelligent agent would need more information to come up with a conclusion about the state of the briefcase. But, because $\Gamma_0 \models holds(up(1), 1)$, the agent knows that currently the first clasp is unlocked.

Consider another example where

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

This history is not consistent with the transition diagram in Figure 8.8 because it has no model. Note that there is no path in our diagram that we can follow in this situation. This makes sense since the briefcase is open iff both clasps are up, but the first clasp must be down at step 1 as the result of $toggle$.

10.2 Defining Explanations

Now let us consider an agent that completed the execution of its n^{th} action. We denote the recorded history of the system up to this point by Γ_{n-1}. In accordance with our agent loop, the agent now observes the values of a collection of fluents at current step n. Let us denote these observations by O^n. The pair $\mathcal{C} = \langle \Gamma_{n-1}, O^n \rangle$ is often referred to as the **system configuration**. If new observations are consistent with the agent's view of the world (i.e., if $\mathcal{C}$ is consistent[2]), then O^n simply becomes part of the recorded history. Otherwise, the agent needs to start seeking the explanation(s) of

[2] Note that syntactically $\mathcal{C}$ can be viewed as the recorded history of the system, but unlike recorded history, it is not yet stored in the agent's memory. However, let us slightly abuse our terminology and talk about the inconsistency of system configuration $\mathcal{C}$ understood as the absence of a model of $\mathcal{C}$ viewed as a recorded history.

the mismatch. As mentioned earlier, the only possible explanation for unexpected observations would be that *some exogenous action(s) occurred that the agent did not observe*. This intuition is captured by the following definition:

Definition 10.2.1. *(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 10.2.1. *(Diagnosing the Circuit-1)*
To illustrate these definitions let us go back to the diagnostic problem from Example 10.0.1. To model the dynamic system from this example, we first need to define its transition diagram. This is done by the following system description, $\mathcal{D}_{ec}$:

Signature

Components:

$$comp(r)$$
$$comp(b)$$

Switches:

$$switch(s_1)$$
$$switch(s_2)$$

Fluents:

$$fluent(inertial, prot(b))$$
$$fluent(inertial, closed(SW)) \leftarrow switch(SW)$$
$$fluent(inertial, ab(X)) \leftarrow comp(X)$$
$$fluent(defined, active(r))$$
$$fluent(defined, on(b))$$

Actions:

$$action(agent, close(s_1))$$
$$action(exogenous, break)$$
$$action(exogenous, surge)$$

Laws

The causal laws, state constraints, and executability conditions describing the normal functioning of our system are encoded as follows:

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

The information about the system's malfunctioning is given by

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

This concludes our system description.

Next consider a history of the system:

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

It is easy to see that the path $\langle \sigma_0, close(s_1), \sigma_1 \rangle$ where

$$\sigma_0 = \{prot(b)\}^3$$

$$\sigma_1 = \{prot(b), closed(s_1), active(r), closed(s_2), on(b)\}$$

is the only model of Γ_0 and hence

$$\Gamma_0 \models h(on(b), 1).$$

In other words, the agent expects the bulb to be lit.

To illustrate the notion of possible explanation, let us now assume that the agent observes that the bulb is not lit (i.e., its prediction differs from reality).

[3] Here, for simplicity, we show only positive literals in the description of states.

In our terminology it means that the configuration

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

is a symptom. As discussed informally in Example 10.0.1 the symptom is expected to have three possible explanations:

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

Two of them are minimal with respect to set-theoretic and cardinality-based orderings of possible explanations. Our formal analysis leads to the same conclusion. 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 (for every $1 \leq i \leq 3$). Hence $\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3$ are the only possible explanations of the symptom. If the agent was to observe that the bulb was OK, $\mathcal{E}_2$ would be the only possible explanation.

It is possible, in fact probable, that the system we would be modeling would be much more complex than the one we have here. Therefore, it is logical to assume that there may be many, possibly irrelevant, exogenous actions in our system. If we were to expand our circuit example by another exogenous action, $make\_coffee$, the addition of this action alone would add three new possible explanations of our original symptom:

$$\begin{aligned}
\mathcal{E}_4 &= \{occurs(surge, 0),\ occurs(make_coffee, 0)\},\\
\mathcal{E}_5 &= \{occurs(break, 0),\ occurs(make_coffee, 0)\},\\
\mathcal{E}_6 &= \{occurs(surge, 0),\ occurs(break, 0), occurs(make_coffee, 0)\}.
\end{aligned}$$

Notice that, although we might want an explanation that includes both $break$ and $surge$, we are probably not at all interested in knowing that some other agent made coffee.[4]

It is clear that no rational agent would be interested in finding all possible explanations of a symptom. Normally, different explanations can be compared with each other using various criteria. This comparison can be represented by an ordering relation between explanations. $\mathcal{E}_1$ can be viewed to be *better* than $\mathcal{E}_2$ if $\mathcal{E}_1$ is a proper subset of $\mathcal{E}_2$, or if the cardinality of $\mathcal{E}_1$ is less then the cardinality of $\mathcal{E}_2$, or if $\mathcal{E}_1$ contains actions that are more

[4] Of course, in the great scheme of things, we may not be able to dismiss this action lightly; it is possible that plugging in the coffee pot might have caused the surge in the first place.

likely to occur or are more relevant to the symptom than those in $\mathcal{E}_2$, and so on. The following definition makes this idea precise.

Definition 10.2.2. *(Best Explanation)*
Let $\mathcal{C}$ be a symptom and $<$ be a partial linear order defined on possible explanations of $\mathcal{C}$ (where $\mathcal{E}_1 < \mathcal{E}_2$ is read as $\mathcal{E}_1$ is better *than $\mathcal{E}_2$). Possible explanation $\mathcal{E}$ of $\mathcal{C}$ is called a* **best explanation** *with respect to $<$ if there is no possible explanation $\mathcal{E}_0$ of $\mathcal{C}$ such that $\mathcal{E}_0 < \mathcal{E}$.*

If the partial linear order on which we base our definition of best explanation is either cardinality-based or subset-based, the best explanations for the circuit problem would be $\{occurs(break, 0)\}$ and $\{occurs(surge, 0)\}$. In some situations one may want to also allow explanation $\{occurs(break, 0), occurs(surge, 0)\}$, but certainly no "best" explanation of our symptom will contain information about occurrences of $make\_coffee$. This result can be achieved by considering explanations consisting of actions "relevant" to the symptom. These actions can be defined as ones that in some way may, directly or indirectly, cause the corresponding fluent (in our case $on(b)$) to become false. Since no law in our system description links making coffee to problems with the circuit, this action can be considered irrelevant. Our best explanations would then be $\{occurs(break, 0)\}$, $\{occurs(surge, 0)\}$, and $\{occurs(break, 0), occurs(surge, 0)\}$ because, by our definition, they are better than the ones containing action $make\_coffee$.

10.3 Computing Explanations

Let us now consider a system with current configuration

$$\mathcal{C} = \langle \Gamma_{n-1}, O^n \rangle.$$

The agent associated with the system just performed its n^{th} action and observed the values of some fluents. Now the agent needs to check that this *configuration is consistent with the expectations* (*i.e., that $\mathcal{C}$ is not a symptom*). As in the case of planning, the problem can be reduced to reasoning with answer sets. To do that we construct program **all_clear** (with parameters $\mathcal{SD}$ and $\mathcal{C}$) consisting of the encoding $\Pi(\mathcal{SD})$ of system description $\mathcal{SD}$, configuration $\mathcal{C}$, and the following axioms:

$$holds(F, 0) \text{ or } \neg holds(F, 0) \leftarrow fluent(inertial, F).$$
$$occurs(A, I) \leftarrow hpd(A, I).$$
$$\leftarrow obs(F, true, I), \neg holds(F, I).$$
$$\leftarrow obs(F, false, I), holds(F, I).$$

with I ranging over $[0, n]$. If program $all\_clear(\mathcal{SD}, \mathcal{C})$ is consistent, then the agent's expectations are consistent with its observations. Otherwise, diagnostics are required.

Each of these axioms deserves some explanation. We begin by briefly introducing them and then elaborating on each one in the examples that follow. The first axiom, sometimes called the **Awareness Axiom**, guarantees that the agent takes into consideration all the fluents of the system. Recall that this is not a tautology. Under the answer set semantics, the disjunction is epistemic, and the statement simply asserts that, during the construction of an agent's beliefs, the value of the fluent in question cannot be unknown; i.e., any answer set of the program must contain either $holds(F, 0)$ or $\neg holds(F, 0)$.

The second axiom establishes the relationship between two similar relations – $occurs$ and hpd. The latter is used to record actions that were actually observed to have happened, whereas the former holds even if the corresponding action is only hypothetical (as, for instance, in planning). As you can see, $occurs$ is a superset of hpd. The rule ensures that, in the process of hypothetical reasoning, we take into account actions that actually happened.

The last two rules, often called the **Reality Check Axioms**, guarantee that the agent's expectations agree with its observations. If they do not, we want the program to realize that there is an inconsistency between what it believes should logically be true versus what it has observed. Note that, despite the apparent similarity between the pair of relations hpd and $occurs$ and that of obs and $holds$, the actual axioms representing these relationships are different. The reason for this difference is discussed in Section 10.5.

The following proposition reduces detection of symptoms to ASP reasoning.

Proposition 10.3.1. *(Symptom Checking)*
A configuration $\mathcal{C}$ is a symptom iff program all_clear$(\mathcal{SD}, \mathcal{C})$ *is inconsistent* (*i.e., has no answer set*).

Example 10.3.1. *(Briefcase)*
Let us again consider the briefcase domain from Section 8.5.1. One can easily check that for configuration $\mathcal{C}_0 = \langle \Gamma_0, obs(open, true, 1) \rangle$ where Γ_0 is

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

program $all\_clear(\mathcal{D}_{bc}, \mathcal{C}_0)$ is consistent, and hence $\mathcal{C}_0$ is not a symptom. If $\mathcal{C}_1 = \langle \Gamma_0, obs(open, false, 1)\rangle$ then $all\_clear(\mathcal{D}_{bc}, \mathcal{C}_1)$ is inconsistent, and as expected, $\mathcal{C}_1$ is a symptom.

Now let us consider the problem of computing explanations of a symptom $\mathcal{C} = \langle \Gamma_{n-1}, O^n\rangle$. Again, this problem can be reduced to computing answer sets of an ASP program. The program, called **diagnose** (with parameters $\mathcal{SD}$ and $\mathcal{C}$), consists of

- the rules of $all\_clear(\mathcal{SD}, \mathcal{C})$
- the **explanation generation** rule

$$occurs(A, I) \text{ or } \neg occurs(A, I) \leftarrow 0 \leq I < n, \\ action(exogenous, A)$$

  or, alternatively,

$$\{occurs(A, I) : action(exogenous, A)\} \leftarrow 0 \leq I < n.$$

- and the following rule, which defines the new relation $expl(A, I)$: The relation holds iff an exogenous action A is hypothesized to occur at I, but there is no record of this occurrence in the agent's history.

$$expl(A, I) \leftarrow action(exogenous, A), \\ occurs(A, I), \\ not\ hpd(A, I).$$

The following proposition reduces computation of a possible explanation of a symptom to extracting atoms formed by relation $expl$ from an answer set of $diagnose(\mathcal{SD}, \mathcal{C})$.

Proposition 10.3.2. *(Computing Possible Explanations)*
A set $\mathcal{E}$ is a possible explanation of a symptom $\mathcal{C}$ iff there is an answer set A of diagnose($\mathcal{SD}, \mathcal{C}$) such that

$$\mathcal{E} = \{occurs(a, i) : expl(a, i) \in A\}.$$

Example 10.3.2. *(Diagnosing the Circuit–2)*
To illustrate the proposition let us again go back to the electrical circuit domain and history Γ_0 from Example 10.2.1. Initially both switches are open, the bulb and the resistor are normal, the bulb is protected, and the agent closes switch s_1. In Example 10.2.1 we showed that a configuration

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

is a symptom with three possible explanations:

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

One can easily check that, using Proposition 10.3.2, these explanations can be extracted from answer sets of program $diagnose(\mathcal{D}_{ec}, C)$. Please see Appendix D.3 for the complete program, $diagnose(\mathcal{D}_{ec}, C)$; we call it `circuit.lp`. Note that, in the presence of complete information for the initial state, it is often convenient to use a CWA to specify the inertial fluents that are false:

```
obs(F, false, 0) :- fluent(inertial,F),
                    not obs(F,true,0).
```

If we had added the above line to `circuit.lp`, then the recorded history could have simply been coded as

```
obs(prot(b),true,0).
hpd(close(s1),0).
obs(on(b),false,1).
```

Defined fluents are taken care of by their CWA.

As expected, invoking this program with, say, `clingo 0 circuit.lp` yields the following:

```
Answer: 1
expl(break,0)
Answer: 2
expl(surge,0)
Answer: 3
expl(break,0) expl(surge,0)
```

Note that the last explanation is not minimal and may or may not be viewed as desirable. At the end of the previous section we discussed an extension of our system by a new action, $make\_coffee$, and showed that it substantially increased the number and length of explanations. All these explanations are found by our method. Possibly the simplest way to limit this number would be to put restrictions on the number of exogenous actions that may be unnoticed by the agent. This can be done by the addition of a simple constraint

```
%% The agent could miss at most one exogenous action.
:- action(exogenous,X1),
   action(exogenous,X2),
   X1 != X2,
   occurs(X1,I),
   not hpd(X1,I),
   occurs(X2,I1),
   not hpd(X2,I1).
```

This constraint eliminates explanation $\{expl(break, 0), expl(surge, 0)\}$, as well as all explanations containing action $make\_coffee$. If we wanted to preserve explanation $\{expl(break, 0), expl(surge, 0)\}$ but still eliminate those that contain irrelevant actions, we could replace the above constraint by

```
:- action(exogenous,X),
   occurs(X,I),
   not hpd(X,I),
   not relevant(X,on(b)).

relevant(break,on(b)).
relevant(surge,on(b)).
```

where $on(b)$ is the fluent whose unexpected value we are trying to explain.

The next section describes other ways of computing best explanations by using existing extensions of ASP. These methods are very similar to those used to find minimal plans in Section 9.5.

10.4 (*) Finding Minimal Explanations

So far we only discussed the way to compute possible explanations of unexpected observations. As seen in the coffee example from Section 10.2, there are normally too many such explanations, often containing completely irrelevant actions. So it would be natural to limit ourselves to computing minimal possible explanations (with respect to cardinality, set-theoretic, or some other type of explanation ordering). Unfortunately, as with the problem of finding minimal plans, there is no known simple way to reduce this to computing answer sets of ASP programs. Therefore, as in Section 9.5 of the chapter on planning, we rely on CR-Prolog or the `minimize` statement.

To reduce the problem of computing minimal explanations to finding answer sets of programs of CR-Prolog, we can simply replace the earlier

explanation-generation rule by cr-rule

$$occurs(A, I) \stackrel{+}{\leftarrow} 0 \leq I < n, \\ action(exogenous, A).$$

The rule says that an observed exogenous action could possibly have occurred in the past, but is a rare event that should be ignored by the agent whenever possible. Accordingly, if a current configuration is not a symptom, these rules are ignored. However, if it is a symptom, then a minimal collection of such rules is used to restore consistency and provide the symptom's diagnosis. (Recall that minimality can refer to set-theoretic or cardinality ordering depending on the options used in the invocation of the CR-Prolog inference engine.) Try adding exogenous action $make\_coffee$ to `circuit.lp`, changing the explanation-generation rule, and running the new program with CR-Prolog.

The `minimize` statement of Gringo that came in so handy for finding minimal plans in Section 9.5 can also be used to compute minimal explanations. Let us consider a program $min\_diagnose(\mathcal{SD}, \mathcal{C})$ obtained by expanding $diagnose(\mathcal{SD}, \mathcal{C})$ with

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

It is clear that the resulting program is going to compute only the diagnosis of $\mathcal{C}$ that is minimal with respect to the cardinality ordering of explanations. Try adding this `minimize` statement to the circuit program that includes exogenous action $make\_coffee$. You will need to use the `--opt-all` option with `clingo` to direct it to return all possible optimizations.

10.5 Importance of New Predicates *hpd* and *obs*

The attentive reader may wonder why our language of recorded history (and consequently the program $diagnose$) uses new predicate symbols obs and hpd instead of the old $holds$ and $occurs$ used in the previous chapter. Of course, part of the explanation lies in these symbol's intuitive meaning. In accordance with assumptions about our agent's ability made at the beginning of this chapter, $obs(f, true, i)$ guarantees the truth of f at step i of the corresponding trajectory, whereas $holds(f, i)$ can also stand for expressing our hypothesis about the value of f at i. (This is exactly how $holds$ was used in planning.) The same goes for hpd and $occurs$. One can still wonder why this distinction is important. What, if anything, will be lost if we stick with our old vocabulary?

To answer this question for obs and $holds$, it is enough to notice that our definition of a symptom depends on the agent's ability to distinguish between observations and hypotheses, and hence, the distinction does not look too surprising. One can still ask, however, why the reality check (which is based on the distinction between observation and prediction) is not written in a way similar to the axiom for hpd and $occurs$ as follows:

$$holds(F, I) \leftarrow obs(F, true, I). \quad (*)$$

$$\neg holds(F, I) \leftarrow obs(F, false, I). \quad (**)$$

To see the difficulty let us consider the following history of some domain

$$obs(f, true, 0)$$

$$hpd(a, 0)$$

where action a does not change the value of inertial fluent f. What would be the value of f at 1? From the observation of f, the awareness axiom, and the original reality check, we have $holds(f, 0)$. From the inertia axiom, the axiom relating hpd and $occurs$, and the independence of f from a, we have $holds(f, 1)$. What would happen now if we were to expand our history by

$$obs(f, false, 1)?$$

Clearly, this observation will contradict our (default) prediction $holds(f, 1)$, and hence the reality check will, as expected, cause a contradiction.

Let's now see how this reasoning would change if we were to replace the reality check by rules (*) and (**) mentioned earlier, which can be viewed as expansions of the agent's definition of $holds$. Suppose there is no observation at step 1, yet f is still true at 0. As before, inertia leads us to believe $holds(f, 1)$. However, the addition of $obs(f, false, 1)$ together with the new axiom (**) allows the reasoner to defeat the inertia axiom and conclude $\neg holds(f, 1)$. This new power of prediction allows the agent to conclude too much – f mysteriously becomes false. This clearly contradicts our general assumption about dynamic domains from Chapter 8 that insists that change is the result of action. Therefore, we defined the reality check in terms of constraints and used the awareness axiom to resolve any possible incompleteness of the initial state.

Summary

In this chapter, we described declarative methodology for solving the basic diagnostics problem – finding possible explanations for discrepancies between what an agent expects and what it observes. As in the case of classical planning, we used a mathematical model that views a dynamic domain as a transition system represented by a system description of $\mathcal{AL}$. The description contains axioms specifying normal behavior of the system together with those describing actions that can cause the system's malfunctioning. These axioms, however, are not enough. In addition, the agent must maintain the recorded history of actions and observations that describes past trajectories of the system possible from the standpoint of the agent. Given this information the agent can determine a symptom – a new observation not compatible with any of these trajectories. The symptom is explained by assuming the existence of past occurrences of exogenous actions that escaped the agent's attention and remain unobserved. The precise definition of a recorded history and its models, as well as the notions of symptom and possible explanation that were presented in this chapter, capture this intuition. To compute possible explanations of symptoms, we expanded our translation of a system description of $\mathcal{AL}$ into logic programs by using several additional axioms including the awareness and reality check axioms. The new translation allowed us to introduce the basic diagnostics module, which is very similar to the simple planning module discussed in the previous chapter. (Instead of hypothesizing about future actions by the agent, the diagnostics module makes hypotheses about past exogenous actions.) The chapter continued with a short discussion of ways to limit our computation to explanations that do not contain irrelevant actions. As in the case of planning, we showed how this can be done in extensions of Answer Set Prolog by consistency-restoring rules and by minimization constructs. The chapter ended with some comments explaining several subtle points related to the language used to record the history of the domain.

The ability to find explanations of unexpected events is another important reasoning task that has been reduced to the computation of answer sets. Even though much more work is needed to fully understand diagnostic reasoning, the material presented in the chapter can serve as the basis for declarative design of intelligent agents. It is especially important to notice that both planning and diagnostic reasoning use the same knowledge of the domain.[5]

[5] Even though this may not be surprising to the readers of this book, historically substantially different knowledge bases were used for each task.

References and Further Reading

The account of diagnosis presented in this chapter is based on Balduccini and Gelfond (2003). The approach has its roots in the early work of Reiter (1987), de Kleer, Mackworth, and Reiter (1992), and Poole (1994), which laid the foundations of logic based diagnosis. Readers interested in learning more about this period may want to consult Hamscher, Console, and de Kleer (1992). Most of the early work on diagnosis was limited to static domains and defined diagnosis in terms of a collection of faulty components that could explain unexpected behavior of the system. In Thielscher (2008) and Baral, McIlraith, and Son (2000), these ideas were applied to dynamic domains, and diagnosis became related to exogenous actions causing the corresponding faults. In Balduccini and Gelfond (2003), the authors simplified these ideas and established the relationship between diagnosis and answer set programming presented in this chapter. Of course our account of diagnostic reasoning only touches on this vast and important area of research. A rather comprehensive panorama of different diagnosis-related problems and solutions can be found in the proceedings of the annual International Workshop on Principles of Diagnosis that started in 1989.

Exercises

1. (a) Give causal laws of $\mathcal{AL}$ for action $receive(X, N, P)$ – person X receives N units of produce P. Use a fluent $has(X, N, P)$ that holds iff person X has N units of produce P.
   (b) Describe a recorded history Γ_1 in which Ray, who initially has three apples, receives two more of them.
   (c) Show logic program $all\_clear(\mathcal{SD}, \mathcal{C})$ for system description $\mathcal{SD}$ from (a) and configuration $\mathcal{C} = \langle \Gamma_1, \emptyset \rangle$.
   (d) Check if $\mathcal{C}$ is a symptom.
2. You are given the following story:
   Mixing baking soda with lemon juice produces foam, unless the baking soda is stale. Joanna mixed baking soda with lemon juice, but there was no foam as a result.
   (a) Represent the story by a system description of $\mathcal{AL}$ and a recorded history.
   (b) Translate this representation into ASP and check if the resulting configuration is a symptom. What knowledge does the agent have about the quality of Joanna's baking soda?

3. You are given the following story:
   Arnold needs to get the book he left in his room. The room is locked and dark, but Arnold has the key and can turn on the light.
   (a) Represent this domain in ASP and write a program that finds a plan that allows Arnold to get the book. *Hint*: Do not forget to tell the planner that one cannot go into a room if one is already in it, etc.
   (b) Suppose that *after Arnold turned on the light switch, the room remained dark*. Add a history to your program that records that the steps in Arnold's plan have been executed and that the room was observed to still be dark. Assume that there are two relevant exogenous actions, $break(switch)$ and $break(power\_line)$.
   Replace your planning module with an explanation module that comes up with a minimal explanation of this unexpected observation.

4. Consider the farmers' market story from Exercise 10 in Chapter 9. Suppose that Fred, Georgina, and Harry are the first customers to arrive at the farmers' market. Fred wants to buy apples, pears, and lettuce. He knows what each of the vendors sells, so he decides to buy apples from Amy first, then lettuce from Bruce, and finally pears from Don. However, when he arrives at Don's booth he learns that Don is out of pears. Write an ASP program that models this domain and its history and comes up with a diagnosis. Note that, because Fred is the one doing the shopping, his actions are considered agent actions, whereas the actions of other customers are exogenous. Natural concurrent actions should be allowed, but obvious restrictions should also apply. To limit possible answer sets, also add the assumption that two customers cannot buy from the same vendor at the same time.

5. You are given the following story:
   Millie is a teenager. She felt fine when she was at school but when she came home she observed that she had a fever and a severe, coated sore throat. She knows that, in teenagers, these are symptoms of both strep throat and mononucleosis. She concludes that she has one of these diseases.
   (a) Represent Millie's medical knowledge in $\mathcal{AL}$ and her knowledge about her actions and observations as a recorded history. Use exogenous actions $infect(strep, Person)$ or $infect(mono, Person)$ that cause a person to have strep or mono, respectively. The symptoms of these diseases can be viewed as indirect effects of these actions.

(b) Translate your representation into ASP, and add the explanation module and rules limiting the number of unobserved exogenous actions to model Millie's reasoning.

(c) *A day later, Millie develops jaundice, which is a symptom of mono, but not strep. She concludes that she has mononucleosis.* Add the new knowledge to you program and run it to find the new explanation. Use action $wait$ to represent that Millie waited a day before observing her new symptom.