Analyzing Spacecraft Configurations Through Specialization and Default Reasoning

Matthew R. Barry
Carlyle M. Lowe

Rockwell Space Operations Company
600 Gemini Ave., R20A-4
Houston, TX 77058

19 January 1990

Abstract

For an "intelligent" system to describe a real-world situation using as few statements as possible, it is necessary to make inferences based on observed data and to incorporate general knowledge of the reasoning domain into the description. These reasoning processes must reduce several levels of specific descriptions into only those few that most precisely describe the situation. Moreover, the system must be able to generate descriptions in the absence of data, as instructed by certain rules of inference. The deductions applied by the system, then, generate a high-level description from the low-level evidence provided by the real and default data sources.

We describe an implementation of these ideas in a real-world situation. The application concerns evaluation of Space Shuttle electromechanical system configurations by console operators in the Mission Control Center. A production system provides the reasoning mechanism through which the default assignments and specializations occur. We provide examples within

this domain for each type of inference, and discuss the suitability of each toward achieving our goal of describing a situation in the fewest statements possible. Finally, we suggest several enhancements that will further increase the "intelligence" of similar spacecraft monitoring applications.

1 INTRODUCTION

This paper addresses the application of default reasoning and specialization techniques toward problems involving pattern classification. A collection of discrete sensor values from a real-time telemetry stream are integrated with certain knowledge about the "world" these sensors represent in order to synthesize an understanding of a situation. By delimiting various intersensor relationships and applying them to subsets of the sensor space, *specializations* of certain situations are achieved. These specializations reduce the number of propositions in the world while maintaining a sort of "semantic equivalence."

In some cases certain sensor values may be missing. This absence of information may be due to some problem, or it may be evidence of a feasible situation in which the *lack* of information is information in itself. For these situations it is reasonable to allow a *default* value for a sensor. This default value may be specified *a priori* or somehow derived from the current context.

1.1 Specialization

Specialization is a fundamental reasoning process employed in *configuration analysis*. By configuration analysis we mean the process of evaluating all of the sensor values within a given context. This can be accomplished by building an evaluation step-by-step from the lowest (least encompassing) statements to the highest (most encompassing) statements. In essence, this process classifies patterns of labelled binary samples into prespecified groups, each of which becomes a sample on its own. The hierarchy of groups represents the specialization of several samples into one *equivalent* sample.

For instance, if

$$\{\phi_1, \ldots, \phi_n\} = P$$

where ϕ_i are labelled samples from context Φ, then

$$\bigwedge_{\phi_i \in P} \Leftarrow \psi_1$$

represents the specialization of the statements ϕ_i into the statement ψ_1. Furthermore, if

$$\{\psi_1\} \cup \{\phi_{n+1}, \ldots, \phi_{n+m}\} = Q$$

then we might apply the successor specialization

$$\bigwedge_{\phi_i \in Q} \Leftarrow \psi_2$$

and so on. For efficiency in our production system implementation (described below), we restrict each sample to one specialization by removing it from the context (database) as it is consumed by the new statement. Considering the last example, we apply

$$\Phi = \overline{Q}(in\Phi)$$
$$\Phi = \Phi \cup \{\psi_2\}$$

after the specialization.

1.2 Default Reasoning

The problem area we consider in this paper belongs to the group of problems in Artificial Intelligence research labelled *common sense reasoning.* In order to draw conclusions based upon certain conditions in an "intelligent"

manner, there must somehow be a higher level of practical information that represents what we might ascribe to a human as "common sense." For example, we might reason "if b is a bird, and we have no reason to believe that b cannot fly, then conclude that it can."

Research efforts related to solving these problems have centered around extending classical mathematical logics to account for implicit information in the database. Typically, this is done by making assumptions about missing information by providing default values. In some cases, providing default values is in itself another problem that must be handled in the reasoning system. Etherington [Etherington 1988] provides a summary of current techniques for handling incomplete information.

1.2.1 The Closed-World Assumption

In an attempt to restrict the reasoning assumptions to information that is available, the *Closed-World Assumption* (CWA) has been developed [Reiter 1978]. The CWA is the assumption of complete knowledge about which positive facts are true in the world. Under the CWA, it is not necessary to explicitly represent negative information. Negative facts may be inferred from the absence of the same positive fact. The CWA corresponds to the knowledge base:

$$\text{if } KB \nvdash P \text{ then infer } \neg P,$$

which states that if the proposition P cannot be derived from the knowledge base KB, then it is reasonable to assume that P is false.

1.2.2 Default Logic

Traditional logics do not possess means for considering the absence of knowledge. Research has considered two sorts of information types whose implementation can extend the capabilities of traditional logics to cover this shortcoming[1]. In the *positive information* category, one assumes that rele-

[1] A formal introduction to default logic may be found in [Besnard 1989].

vant information is known, therefore anything that is not known must be false. In the *default information* category, one has default values available to fill gaps in the absence of specific evidence. It is this *default information* category that describes the reasoning process embodied by our classifier.

A *default logic* may be constructed from a standard *first-order logic* by permitting addition of new inference rules [Reiter 1980,Reiter and Criscuolo 1981]. These new rules allow *known* and *unknown* premises, making possible conclusions based on missing information. A *default theory*, Δ, is an ordered-pair (D, W) consisting of a set of *defaults*, D, and a set of *first-order formulae*, W. The fundamental statements in Δ are *defaults*, defined by the expression:

$$\frac{\alpha(\overline{x}) : \beta_1(\overline{x}) \ldots \beta_m(\overline{x})}{\gamma(\overline{x})}$$

where $\alpha(\overline{x})$, $\beta_i(\overline{x})$, and $\gamma(\overline{x})$ are formulae whose free variables are contained in $\overline{x} = x_1, \ldots, x_n$. This expression states that if certain *prerequisites* α are believed, and it is consistent to believe that certain *justifications* β are true, then it is reasonable to sanction the *consequent* γ [Etherington 1988]. If $\beta(\overline{x}) = \gamma(\overline{x})$, then the default is *normal*. If $\beta(\overline{x}) = \gamma(\overline{x}) \wedge \omega(\overline{x})$, for some $\omega(\overline{x})$, then the default is *semi-normal*.

This capability to withdraw a previous assumption and reconstruct a new set of conclusions is known as *nonmonotonic reasoning* [Ginsberg 1987].

2 Application

The application we present involves the operational evaluation of Space Shuttle electromechanical component configurations by flight controllers in the Mission Control Center (MCC). Specifically, specialization and default reasoning techniques have been applied to one of the tasks involved in monitoring two Shuttle propulsion subsystems: the *Orbital Maneuvering System* (OMS) and the *Reaction Control System* (RCS).

2.1 Overview

To operate the OMS and RCS, Shuttle astronauts manipulate a collection of switches controlling valves that direct the fluid flows throughout a plumbing network. Many of these switches control two valves simultaneously: an oxidizer system valve and the corresponding fuel system valve. Position indicators within the valves and switches provide insight into their mechanical position.

Flight controllers in the MCC help the astronauts to manage these systems by monitoring the on-board configuration. The information available to the flight controllers is more complete than the information available to the astronauts Valve and switch positions appear to the flight controllers as binary values noting presence of (or lack of) an open indication, close indication, or both. A set of 16-bit *configuration words* relay all of the available measurements to the flight controllers.

The MCC computers help the flight controllers to monitor the on-board valve and switch configuration by executing a program that compares *actual* and *expected* configurations. Since only some of the bits in a given configuration word apply to the systems of interest, the comparison procedure includes a set of masking words. When the bit patterns that are not filtered by the mask disagree, the program indicates a problem by displaying a certain status character next to that word. Since the contents of those words are displayed in hexadecimal notation, the operators are made aware of a discrepancy condition through this status character, but are not informed of the specific discrepancy. Furthermore, several discrepancies may occur in the same word.

The process of manually deciphering the hexadecimal information is time consuming and prone to error, so we use a computer program to decode any word of interest. This program displays English descriptions of the indications corresponding to those bits that do not match the expected bit pattern. It is up to the operator, however, to remember the patterns from each individual decoding, and to construct a complete signature interpretation from several hexadecimal words simultaneously.

2.2 Reducing Information

The classifier we describe was developed to perform this decoding and *signature construction* task through belief specialization and default reasoning. The decoding program was extended to isolate each bit in the configuration words and to generate a *statement* for a database describing the observed and expected indications. The classifier then attempts to generate a state description for these indications. The state descriptions offer an explanation in high-level, intuitive, terminology. For example, instead of being offered the four statements

```
Open(ox-valve,manifold-1)
Open(fu-valve,manifold-1)
¬Closed(ox-valve,manifold-1)
¬Closed(fu-valve,manifold-1)
```

the flight controller is informed

```
Open(valves,manifold-1)
```

due to the application of a typical specialization rule

```
Open(ox-valve,x) ∧
Open(fu-valve,x) ∧
¬Closed(ox-valve,x) ∧
¬Closed(fu-valve,x) ⇒
Open(valves,x)
```

where `x` is bound to `manifold-1`. Better still, if the database includes the statements

```
Open(valves,manifold-1)
Open(valves,manifold-2)
Open(valves,manifold-3)
Open(valves,manifold-4)
Open(valves,manifold-5)
```

then the best description is

```
Open(valves,all-manifolds)
```

from the specialization

```
Open(valves,manifold-1) ∧
Open(valves,manifold-2) ∧
Open(valves,manifold-3) ∧
Open(valves,manifold-4) ∧
Open(valves,manifold-5) ⇔
Open(valves,all-manifolds)
```

Carrying on to "meta-level" statements regarding a "configuration of configurations," one might make the specialization of the statements

```
Open(valves,all-manifolds)
Open(valves,rcs-regulators)
Open(valves,loms-crossfeed)
Open(valves,all-prop-tanks)
On(heaters,thrusters)
Off(heaters,pods)
⋮
```

resolve to the implicit description

```
Configuration( Prelaunch )
```

2.3 Missing Information

One important consideration in the problem is that *lack of evidence regarding a position indication is important information.* That is, missing information may imply a certain position indication. For the OMS and RCS, this is the case with the switch positions: lack of an OPEN or CLOSED

indication means that the switch is assumed to be in the GPC (General Purpose Computer) position for computer-controller valve operation. This corresponds to the semi-normal default rule (without prerequisites)

$$\frac{: \mathtt{GPC(s)} \wedge \neg \mathtt{Open(s)} \wedge \neg \mathtt{Closed(s)}}{\mathtt{GPC(s)}}$$

for switch **s**.

Missing information is also important in valve positions. Many valves lack instrumentation of the CLOSED position, so if the OPEN indication is not present, then one must assume that the valve is closed. Similar to the switch position default, this corresponds to the semi-normal default rule

$$\frac{: \mathtt{Closed(v)} \wedge \neg \mathtt{Open(v)}}{\mathtt{Closed(v)}}$$

for valve **v**.

3 Implementation

The sort of reasoning process described above can be implemented through the use of a commercial production system. Statements providing a *specialization of beliefs* conveniently can be represented as conventional production rules. The left-hand side of the rule consists of one or more statements which, when considered together, imply a more specialized statement having equivalent meaning. The right-hand side of the rule asserts the consequent statement and retracts all of the prerequisites that were held true in order to fire the rule. This process decreases the total number of statements in the database, while maintaining equivalent knowledge within the reasoning world. The system can retract its own conclusions (and assumptions) later in the deduction process, thereby exhibiting nonmonotonic reasoning.

3.1 Design

The application we describe uses a combination of procedural and declarative programming techniques. NASA's C Language Integrated Production System (CLIPS) provides rule processing capabilities. A host program, written in C, acquires the necessary data and applies a valuation algorithm to generate statements (facts) for the database. This algorithm assigns to each positive component position indication a description of the component, a description of the position indication (e.g. **Open**, **Closed**, **On**, or **Off**), and a qualifier as to whether that position belongs to the *actual* or *expected* configuration. When all necessary statements have been generated, the production system evaluates them and builds the state description with the given inference rules. The contents of the database after all inferences have been performed (i.e. when no more rules fire) represents the conflict set between the actual and expected configurations. The host program translates this set of statements into English sentences for display to the operator.

Since the independence of valve or switch state indications is not guaranteed by the physical system, so this independence is not required by our production system. That is to say, though the valves are intended to reside in either the opened or closed states, the indications may not provide conclusive evidence and perhaps no default assumptions are available. For these situations none of the statements that consider the guilty valve will be applied, thereby leaving the lowest level samples in the database. This is a desirable characteristic of the program, causing it to provide all of the evidence that was not reduced through the inference process. Moreover, facts are held based on observed world states rather than assumed states[2].

In order to reason about defaults one must be able to decide when information is missing. Our application uses the CLIPS *not* operation for this purpose. This operation evaluates to TRUE if a match is *not available* for the pattern, thus allowing us to determine that default-overriding evidence is not present in the database. If the default indication is the only one

[2]There remains the underlying assumption, however, that the observed state represents the actual state.

available for a particular sensor, then the value provided as the default value for that sensor becomes the value of the missing pattern. If any evidence other than the default value is available, that evidence is used in the classification process. These *default processing rules* fire first so as to build all of the lowest-level indications before starting specializations. A typical default rule looks like this:

```
(defrule expect-switch-defaults
 (default ?dom ?item
      ?d&sp-op|sp-cl|sp-gp)
 (not (actual ?dom ?item sp-op))
 (not (actual ?dom ?item sp-cl))
 (not (actual ?dom ?item sp-dm))
 (not (actual ?dom ?item sp-gp))
=>
 (assert (actual ?dom ?item ?d))
)
```

This rule extracts a default indication from the default table, specifying that it handles only switches by restricting the pattern match to one of the three reasonable switch values (the value of *dilemma* (**sp-dm**), though a possible observed state, is not a reasonable default value). It then proceeds to search for an overriding indication by looking for all possible switch values in the **actual** indications. If a match is found, then an **actual** indication is present and the rule fails. If no match is found then the default value is appropriate, so the rule fires, asserting the default value as the **actual** value.

Most of the production rules in our application represent the *specialization rules*. These rules assemble collections of patterns into a more specialized pattern implying the same information. The right-hand side of the rule retracts the premises and asserts the conclusion. Each of these rules works for either of the two comparison states. Recalling the manifold example provided earlier we demonstrate a specialization rule as shown below. This rule collects all five of the named manifolds for an arbitrary domain **dom** and either specialization mode (**actual** or **expect**). Providing the switch

and valve positions (?s and ?v) for each manifold are the same, the rule asserts the special conclusion ?dom manifolds. Prior to the special assertion, however, the rule retracts the prerequisites from the database[3].

```
(defrule specialize-group-manifolds
 ?m1 <- (?mode&actual|expect
         ?dom manifold-1 ?s ?v)
 ?m2 <- (?mode
         ?dom manifold-2 ?s ?v)
 ?m3 <- (?mode
         ?dom manifold-3 ?s ?v)
 ?m4 <- (?mode
         ?dom manifold-4 ?s ?v)
 ?m5 <- (?mode
         ?dom manifold-5 ?s ?v)
=>
 (retract ?m1 ?m2 ?m3 ?m4 ?m5)
 (assert(?mode
         ?dom manifolds ?s ?v))
)
```

4 Extensions

Though the techniques we have employed constitute a powerful application, there are a variety of enhancements that can be made to the reasoning process. We outline a few of them here.

4.1 Temporal Reasoning

Comparing an *actual* signature with an *expected* signature can sometimes be interpreted as a matter of temporal persistence. If we can make assumptions

[3]The retraction is performed before the assertion to minimize the complexity in driving new patterns through the network.

about the dynamic behavior of the measured system, then we can draw from knowledge of the *expected* state to help make assumptions about the *actual* state.

One can imagine running a configuration evaluator continuously (ours runs only upon demand), focusing only on those indications that change in the signature. An interesting enhancement therefore might be in *predicting the next signature* by incorporating knowledge of procedures and time [Georgeff 1987].

4.2 Analog Reasoning

Though the information provided as input to the classifier currently is discrete (binary), analog information may also be important in describing a configuration. For example, some valves may not have discrete position indications, but rather "percentage open" indications. There may be guidelines for interpreting "percentage flow" through these valves that could be implemented as rules with thresholds on their left-hand sides. If a valve is indicating 2% open, for example, the interpretation will probably lead to considering this valve closed.

Analog interpretations may also be used to reason about system measurements that are not strictly part of the "configuration." We might include considerations for thermodynamic measurements in our evaluation, building flow hierarchies, limit violation detectors, or determining "degrees of wellness" for analog components.

4.3 Evidential Reasoning

A variety of problems may be introduced into the classification process by supplying nonrepresentative signatures as input. There are many orbiter component failures that will cause an invalid signature to be relayed to Mission Control. For example, failure of a computer, demultiplexer, signal conditioner or transducer will cause all of the telemetry measurements associated with that components to be incorrect, without affecting operation

of the measured device. These conditions are detectable, however, and can be provided as input to the classifier. When the classifier is made aware an instrumentation component failure, and it "knows" the measurements derived from that component, then it can take this invalid information into account when performing the classification.

Sometimes the instrumentation failure may not be known before a classification process begins. In these cases it might be useful to refer to *sub-signatures* that one can map onto the actual signature, measuring the degree to which each body of evidence supports the indicated signature. The heuristics for interpreting competing signatures will likely involve *evidential reasoning* [Lowrance 1986].

5 Summary

The motivation behind this project has been to desire to demonstrate the capabilities of applied default reasoning and specialization as realized through a typical production system. We described a system that implements these reasoning paradigms in a real-time telemetry monitoring application. This application performs a complete task, relieving flight controllers from this duty and allowing them to address their attention to other activities. Due to its declarative construction, the system is able to accommodate changes in the "world" without restructuring the inference process. Most importantly, the system is able to perform a mundane task frequently, consistently, and inexpensively, while producing expert-level results.

We also described several enhancements that seem to be logical extensions to the current system. These extensions will be investigated in the near future.

References

[Besnard 1989] Besnard, *An Introduction to Default Logic*, Springer-Verlag,

Berlin, 1989.

[Etherington 1988] Etherington, *Reasoning with Incomplete Information*, Morgan Kaufmann Publishers, Inc., Los Altos, CA, 1988.

[Georgeff 1987] Georgeff and Lansky, "Procedural Knowledge," SRI International Technical Note 411, Menlo Park, CA, 1987.

[Ginsberg 1987] Ginsberg, *Readings in Nonmonotonic Reasoning*, Morgan Kaufmann Publishers, Inc., Los Altos, CA, 1987.

[Lowrance 1986] Lowrance, Garvey and Strat, "A Framework for Evidential- Reasoning Systems," in *Proceedings of the Sixth National Conference on Artificial Intelligence*, 1986.

[Reiter 1978] Reiter, "On Closed-World Data Bases," in *Logic and Data Bases*, Gallaire and Minker (eds.), Plenum Press, New York, 1978.

[Reiter 1980] Reiter, "A Logic for Default Reasoning," *Artificial Intelligence 13*, North-Holland, 1980.

[Reiter and Criscuolo 1981] Reiter and Criscuolo, "On Interacting Defaults," *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, 1981.