Technical Report

UCAM-CL-TR-214
ISSN 1476-2986

Number 214

UNIVERSITY OF CAMBRIDGE

Computer Laboratory

Integrating knowledge of purpose and knowledge of structure for design evaluation

J.A. Bradshaw, R.M. Young

February 1991

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500

https://www.cl.cam.ac.uk/

Technical reports published by the University of Cambridge
Computer Laboratory are freely available via the Internet:

https://www.cl.cam.ac.uk/techreports/

ISSN 1476-2986

Integrating Knowledge of Purpose and Knowledge of Structure for Design Evaluation

J. A. Bradshaw
Computer Laboratory
University of Cambridge
New Museum Site
Cambridge, CB2 3QG
email: john.bradshaw@cl.cam.ac.uk

R. M. Young
MRC Applied Psychology Unit
15,Chaucer Road
Cambridge, CB2 2EF
email: richard.young@mrc-apu.cam.ac.uk

February 1, 1991

to appear in IEEE Expert (April, 1991)
as part of the lead set of papers for the special track on Functional Reasoning.

Abstract

This paper describes a knowledge representation strategy, for mechanical devices, which combines *Knowledge of Structure* and *Knowledge of Purpose. Knowledge of Purpose* specifies how devices are expected to behave and *Knowledge of Structure* details how devices are connected. Knowing 'correct' behaviour (*Knowledge of Purpose*) it is possible to usefully comment on any generated behaviour, predicted or actual. Generation of behaviour is a bottom up process (from components to systems) whereas behaviour evaluation is top down (from systems to components). Common purpose is used to group devices into systems.

The core evaluation activity is the generation of an envisionment graph (similar to that described by deKleer and Brown [deK84]). The complete graph represents the full set of predicted behaviour states for the represented device. These behaviour states are compared with the *Knowledge of Purpose* behaviour descriptions; if conflicts are found then these are described and the structure and purpose descriptions of the device are scanned to establish the source of the conflict. The ideas discussed in this paper are implemented in the Doris system which is described.

1 Introduction

The design of mechanical devices usually follows a particular, describable pattern. First the intended, correct behaviour of the device is described: this is the purpose which the device should achieve. From the purpose description a structural description of a device which will achieve the purpose is generated. Without both Knowledge of Purpose and Knowledge of Structure it is not possible later to perform some of the most basic tasks. For example, if only Knowledge of Purpose is held then it is possible to comment on the actual behaviour of a real device but it is not possible either to predict behaviour or to locate the source of a conflict when incorrect behaviour is detected. Having only Knowledge of Structure it is possible to predict behaviour but it is impossible to judge this or actual behaviour. Therefore it is important that both types of knowledge are represented if a useful device representation strategy is to be developed.

A representation strategy which holds Knowledge of Purpose and Knowledge of Structure has been developed. The purpose information, which drives the design task, is represented separately from the structural information. This separation is explicit because of the advantages which accrue both at the behavioural level and at the structural level.

At the behavioural level concentrating on describing purpose provides a clean description of correct behaviour. Knowing correct behaviour enables us to comment on generated behaviour (predicted or actual) of a device, whatever the behaviour. If the resultant behaviour is incorrect (*ie* it conflicts with the purpose) then it is the way in which this behaviour deviates from the purpose (the correct behaviour) which is both interesting and important. If there is no conflict, the purpose description is then also the behaviour description.

There is no need to describe both incorrect and correct behaviour since the one is the complement of the other: by knowing one the other is also known. Having to describe only the correct behaviour is a significant positive feature of this system since it is very much simpler and more efficient to describe how a device should behave than to have to try to predict and describe the full range of incorrect behaviours.

At the structural level we model devices so that the model behaves in a manner similar to actual devices. Actual devices know nothing of the system around them and it is only through their connections that their behaviour is effected and that they influence the wider world, *ie* a device's behaviour is a function of its inputs. From this a notion of the behaviour of a generic device has been described in terms of the state of its inputs. Thus the behaviour of any device can be predicted by knowing its connections and its inputs. The structural description is *device centered* in that for every device only local information is held. This approach has had two important benefits. First it keeps local information local, *ie* in order to describe one part, knowledge of any other part of

a system need not be known or understood, and secondly a library of standard parts can be used in much the same way as an engineer draws on standard parts from his workshop to assemble a particular system.

This paper concentrates on describing the representation introduced above as it relates to the design evaluation task in particular. The design evaluation task checks that the structural design describes a device which will achieve its stated purpose. For simple systems design evaluation is a simple task. It is possible for an individual to conceptualise all the different scenarios which could affect a simple device (*eg* a battery, switch and light system) and fully understand how it would behave under these different scenarios. However with even a small increase in complexity understanding design evaluation rapidly becomes a non trivial task. The ideas expressed in this paper are intended to provide a means by which this aspect of device understanding can be automated, thus providing a systematic mechanism for evaluating the logical aspects of the design of mechanical causal systems. These ideas described here have been implemented in the Doris system, as has the example presented.

The development of the ideas embodied in the Doris system follows from the work done by the Qualitative Reasoning community [deK84, For84, Kui84], particularly the *component centered* approach described by deKleer and Brown [deK84]. We have found the notions of mythical and normal time very useful but do not follow their *No Function in Structure* principle. At the structural level local knowledge is kept local but the purpose descriptions are not restricted in the same way. It is the context in which a device exists which affects its behaviour. It is, therefore, not possible to describe behaviour without using knowledge of its context. Mythical and normal time are simply means which enable devices, which operate in parallel, to be modelled by a sequential processor. We do not advance time by set time units but follow the approach used in QSIM [Kui86] of advancing time by a variable amount sufficient to allow an interesting qualitative change to occur. We make use of qualitative values and reasoning in the Doris system. There has been some expansion in the flexibility of these values to allow a description of their type and range as well as the relationship between the values within the set. This gives the designer more freedom to describe precisely the values which are significant in his system. However, we do not describe purpose in terms of qualitative differential equations; rather we have chosen to describe purpose im a manner similar to the device understanding work of Sembugamoorthy and Chandrasekaran [Sem86]. We construct an envisionment graph which is similar to that described by deKleer and Brown. The envisionment graph is a directed graph where the nodes represent different behavioral states and the arcs represent behaviour state transitions. The envisionment graph is complete in that all reachable behaviour states are represented and no unattainable behaviour states are included.

As already stated above we have also been influenced by the 'Device Understanding' community,

particularly by the functional representation work of Chandrasekaran and his colleagues [Sem86, Cha86, Keu89, Goe89]. However, rather than beginning with descriptions of the device's actual behaviour we generate the behaviour states and their transitions in an envisionment state graph. Purpose describes the desired behaviour rather than the actual behaviour. We therefore provide a 'lower level', more static description of the device specifying only its structure and its purpose. It is the analysis of the envisionment graph, together with the Knowledge of Structure and Knowledge of Purpose which enables us to perform the task of design evaluation. The advantages which are to be had over the functional representation work is that the knowledge which needs to be captured is articulated during the design process and remains constant for the duration of the life of the device, except of course during redesign. This means that when a device begins to behave incorrectly, perhaps due a to broken component, no new behaviour description need be externally described. Only the status of the observable variable values are required. Using these values and the device representation the Doris system can describe the nature and location of the fault.

The important points described here are:

- That *device centered*, structural knowledge provides a good basis for predicting behaviour.
- That Knowledge of Purpose describes the intended and 'correct' behaviour of the device and can be used to evaluate generated behaviour - predicted or actual.
- That the manner in which Knowledge of Purpose is used removes the need to describe 'incorrect' behaviour.
- That this representation strategy provides a useful basis which will support several different applications without requiring any alteration of the device descriptions.

In order to illustrate the ideas presented in this paper we shall use a model of a simple greenhouse heating system. The purpose of this system is to maintain a higher than ambient temperature in a greenhouse during particular periods. The system consists of an electrical timer and an environment control system. The environment control system itself consists of a thermostat and a heater. Figure 1 shows how this system is connected. This example has been implemented in the Doris system and the results presented are those generated by Doris.

Obviously there is a problem with using a simple system for the purposes of illustration. The intent of the ideas described here is that the device to be evaluated should be more complex than can be comfortably handled by an individual. However, this simple greenhouse system is used to illustrate the ideas rather than as an example of the sort of complexity which Doris is intended to handle. The most significant system, for example, which has been considered thus far is the hydraulic system for the British Aerospace 146 passenger jet. This model is still under development. This system has a high degree of redundancy (three pressure systems) and consists of many devices (>40

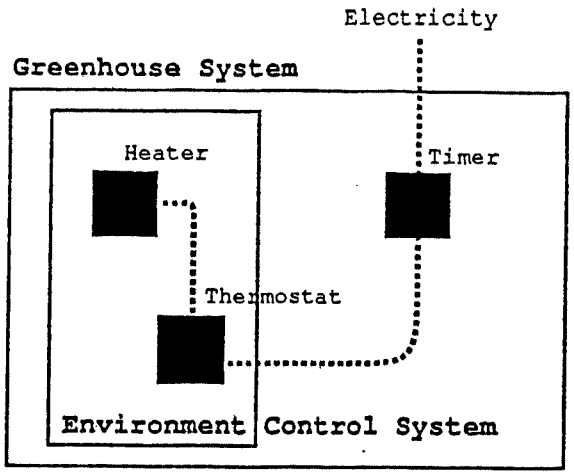


Figure 1: Structure of the Greenhouse System

important devices) and many variable attributes (pressure, flow rate and fluid level are the most important of these but the relationship between the position and settings of the spoilers, rudder *etc* are also monitored).

This paper is organised along the following lines. The next two sections describe the nature and representation of Knowledge of Structure and Knowledge of Purpose. Following that is a description of the task which Doris has been designed to execute. The final two sections consider other applications (simulation, fault diagnosis and explanation) and discuss the results generated from this work.

2 Knowledge of Structure

2.1 Devices

Knowledge of Structure is simply the knowledge of what devices form the overall system and how they are connected. A device is either a system or a component. A system device consists of other systems and components. Components are simple, indivisible devices.

All devices share a common structural framework, see Figure 2. They receive inputs and produce outputs via named ports. The originating device and output port from which the input is sourced is declared with each input port. A device may also embody a certain amount of logic, which we call *control*. At any time, a device is in one of three states - quiet, alert or energetic - and its behaviour is a function of its inputs and control. The device is quiet when its inputs are not available, alert

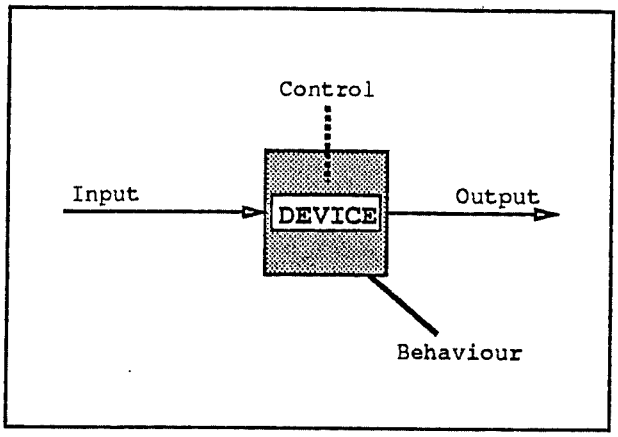


Figure 2: Device structure

when its inputs are available but control resolves to *not true*, and energetic when its inputs are available and its control resolves to *true*. The device as described here is what we regard as the *generic device.*

Devices are connected together by connecting their input and output ports. Every device also has associated with it a purpose. The actual purpose description is held separate from the structure description for reasons which are discussed in the following section. Within the structural description there is only an identifier which provides the link between the structural description and a named purpose description.

The actual representation of the greenhouse system's thermostat is shown in Figure 3.

```
device(d_greenhouse_thermostat, component,
       p_greenhouse_thermostat,
       control([equal(v_greenhouse, temp, low)]),
       input([port(elec, d_environ_control, elec)]),
       output([port(e_out)])).
```

The name of the device is 'd_greenhouse_thermostat'. It is a 'component' and has a purpose 'p_greenhouse_thermostat' which is defined elsewhere. This device also embodies some internal control logic - 'equal(v_greenhouse, temp, low)'. It has a named input port - 'elec' - which is connected to the 'elec' port of the device 'd_environ_control'. It has one output port - 'e_out'. The thermostat is energetic when the temperature is low and when its electrical input is available.

Figure 3: A Component Device

Our use of the term *structure* is perhaps narrower than that normally employed. We do not represent each device's behaviour explicitly, but by knowing the behaviour of the generic device we are able to determine an actual device's behaviour given the status of its external influences. Thus our *knowledge of structure* holds specific physical information about particular instances of devices which allows their behaviour to be determined, as dictated by their external influences and internal

logic.

2.2 Variables

In addition to the structure of devices we need also to model the changes of the interesting variables in a system. In the greenhouse system we are interested in temperature (in this example called 'v_greenhouse, temp') and time ('v_clock, time'). In addition to specifying the way in which values change, we also need to represent variable type, initial value and whether the value is generated internally or externally. Variable type is either qualitative, quantitative or binary. Figure 4 presents the information held for the greenhouse temperature attribute.

```
variable(v_greenhouse, temp, qualitative, low, internal,
        logic((energetic(d_greenhouse_heating)),
                increase_value(v_greenhouse, temp),
                decrease_value(v_greenhouse, temp))).
```

'v_greenhouse, temp' is a qualitative variable. It has an initial value (for the purposes of envisionment) of 'low'. Its value is affected by devices 'internal' to the greenhouse system thus it is possible to determine how the value will change solely by understanding the relationship between it and the devices within the greenhouse system and without any need to consider possible external influences. The 'logic' structure should be read as an *if-then-else* statement. When the device 'd_greenhouse_heating' is energetic then 'v_greenhouse, temp' will increase else it will decrease.

Figure 4: An internal, qualitative variable

For qualitative variables we have also to specify which values we are concerned with and how they are related to each other. For example, Figure 5 shows this information for 'v_greenhouse, temp'. It is possible to define qualitative variables where there is no distinction made between the decreasing, steady and increasing states or where these are only significant for the intermediate values, *ie* not for those at the extremes.

```
range(v_greenhouse, temp, [low, normal, high], [r,r]).
```

'low' is less than 'normal' which is less than 'high'. The fourth parameter is used to describe the nature of the qualitative values. A list indicates that the values have distinct increasing, decreasing and steady components to them. The identifiers in the list may be either 'p' (point) or 'r' (range), these are used to give particular information about the character of the lowest and highest qualitative values. 'p' indicates that the associated end value's component parts are not significant and 'r' that they are. A 'one' in place of the list would have indicated that only the actual value without the direction of movement is significant.

Figure 5: Qualitative variable range

It can be useful to know the relative time taken for different variables to change value. It can be specified that variables will change in an order of magnitude faster 'higher_order' or with the same order of magnitude 'same_order'. Figure 6 shows how the fact that within the greenhouse system it is expected that temperature will change faster than the value of 'v_clock, time'.

```
higher_order((v_clock, time), (v_greenhouse, temp)).
```

'v_greenhouse, temp' changes status an order of magnitude faster than 'v_clock, time'. This means that between changes in 'v_clock, time', 'v_greenhouse, temp' should have the opportunity to change at least once. 'v_greenhouse, temp' may in fact maintain the same value but nevertheless it must be recomputed before 'v_clock, time' is again considered by the evaluation system.

Figure 6: Relative time for variable change

3 Knowledge of Purpose

Knowledge of Purpose is associated with each device. Knowledge of Purpose specifies a device's intended behaviour and so provides the basis on which to evaluate either its actual or predicted behaviour. It is important to emphasise that *purpose* specifies intended behaviour and does not describe actual behaviour, actual behaviour being generated during envisionment. The structural representation of the thermostat was shown in Figure 3, it referred to a purpose description 'p_greenhouse_thermostat'. This purpose description appears separately from the structural description of the device (see Figure 7 for 'p_greenhouse_thermostat' description). This feature mirrors our perception that structure and purpose descriptions are the product of different stages in the overall design process. Even during redesign it is possible to see how changes which are made to purpose descriptions are handled separately from changes made to the structural design.

```
purpose(p_greenhouse_thermostat, weak,
        what([keep(v_greenhouse, temp, normal)]),
        when([active(d_greenhouse_thermostat)])).
```

The purpose 'p_greenhouse_thermostat' is satisfied when the device 'd_greenhouse_thermostat' is active (*ie* alert or energetic) and the variable 'v_greenhouse, temp' is normal. The *strength* of the purpose logic is 'weak' which signifies that the *what* must be true if the *when* is true but if the *when* is not true then no judgement is made about the status of the *what logic*. If rather than 'weak' the strength were 'strong' then either the *what* must be true when the *when* is true or the *what* must be false for the purpose to be satisfied.

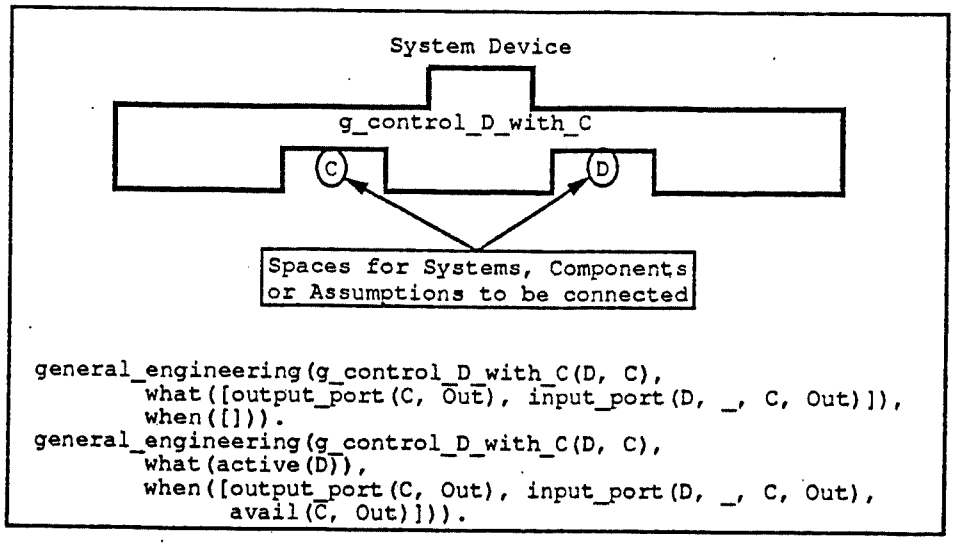
Figure 7: Purpose description for the greenhouse thermostat

A vocabulary of terms such as 'keep' and 'active', for the purpose of evaluating the status of variables and devices, has been defined. *Keep* is satisfied if the variable holds the desired value or if it is changing towards this value. Since we know the range of values for the variable and how they are related to each other it is possible to check the direction of change. A device is active if its control logic resolves to true; it is thus active when it is either alert or energetic, *ie* when it is not quiet. The flexibility of the representation allows us to define device specific terms if required, as well as to alter the default descriptions provided.

Knowledge of Purpose also provides the basis on which several devices are grouped together to

form a system - a complex device. Recognising this feature of systems enables us to develop hierarchically oriented representations of complex devices. The advantage of this approach is that for very large and complex systems we are able to examine parts without being forced to model the whole. Thus if a system A is made up of sub-systems I, J, and K and we know that these sub-systems behave as expected, they may be represented as simple, component devices for the purpose of describing A as well as for examining the behaviour of A. This is a completely natural approach, we do not describe the behaviour of a power station starting with descriptions of the smallest components. Rather we describe its purpose and, if required, how its different sub-systems combine to achieve this. If more detail is required we explain how these sub-systems behave in terms of their purposes and their parts. In the Greenhouse system, the fact that the timer and the environment control system combine to achieve the greenhouse system's purpose enables us to group them together into a named, complex device.

3.1 General Engineering Techniques




```
general_engineering(g_control_D_with_C(D, C),
        what([output_port(C, Out), input_port(D, _, C, Out)]),
        when([])).
general_engineering(g_control_D_with_C(D, C),
        what(active(D)),
        when([output_port(C, Out), input_port(D, _, C, Out),
              avail(C, Out)])).
```

This template represents an arrangement of devices where a device D is controlled by controlling one of its inputs using a device C. The devices D and C can themselves be either systems, components or assumptions.

Figure 8: General engineering template

Common purpose allows us to group components together and leads to the last important type of knowledge which we consider and represent. The way in which devices are connected influences the manner in which the whole system will behave. In certain instances the manner of their interconnection follows some generally accepted principle. For example, an accepted way of controlling the behaviour of a device is to control one or more of its inputs (the actual representation is shown at the bottom of Figure 8). We have taken this notion of 'general engineering techniques'

and included it within the description of system devices.

General engineering templates provide designers with an arrangement of devices which is already understood. The greenhouse heating system, as shown in Figure 9, makes use of two templates. The first is that we can control the environment control system by controlling one of its inputs (electricity) by using a timer. The other template used embodies the knowledge that a variable (temperature) can be kept constant if there is a device (the thermostat) which will measure its value and if there are two devices which will respectively raise and lower the temperature. Time and temperature are variables which influence the behaviour of the devices 'Timer' and 'Thermostat' respectively.

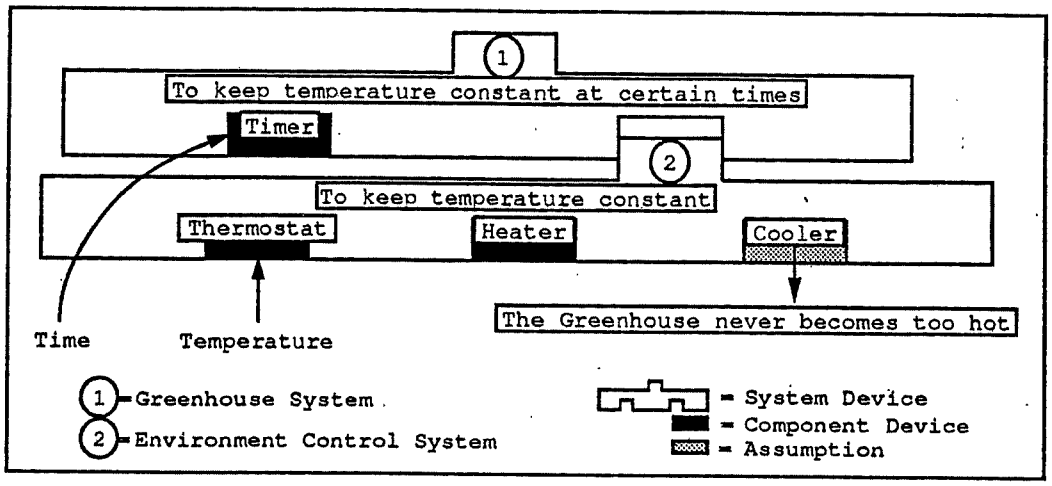


Figure 9: Greenhouse system engineering

The fact that in this system the cooling system is represented by an assumption rather than by an actual device is discussed in the next section.

3.2 Assumptions

Apart from recording the purpose of devices which form part of a system the designer may sometimes incorporate assumptions into the system's design. In understanding how the device should behave and trying to explain the reasons for any deviation from this expectation it is often useful to have these assumptions explicitly recorded. The overall purpose of the greenhouse system is to maintain a higher than ambient temperature at certain times of day. Thus the expectation is that the greenhouse need never be cooled and so no cooling component included, only a heater. Therefore the cooling device is replaced by the explicit assumption *the greenhouse never becomes too hot* (as shown in Figure 9). For the Greenhouse system to behave correctly this assumption

must always hold true. The logic for assumptions (as illustrated in Figure 10)works in the same way as that for purpose descriptions.

```
assumption(d_greenhouse_cooling,
        what([decreasing(v_greenhouse, temp)]),
        when([not(energetic(d_greenhouse_heating))])).
```

When 'd_greenhouse_heating' is not energetic then 'v_greenhouse, temp' should be decreasing.

Figure 10: Cooling assumption

Comment

Knowledge of Purpose is the basis on which the structural design is developed. The designer's task is to produce a structural design which satisfies the specified purpose. The specification of the structural description and the purpose description are the results of different phases in the design process, even if they both come from the same source. During the structural design it often becomes apparent that certain objectives are unattainable and then either the intended purpose must be modified or the structural design changed. Even then the process of formulating a different structural design is distinct from the purpose modification process. Therefore we found it natural to clearly separate (physically as well as logically) Knowledge of Purpose from Knowledge of Structure.

4 Doris

The object of the Doris system is to evaluate the behaviour of devices, checking that their computed behaviour does not conflict with their specified purpose.

We have followed the approach of deKleer and Brown [deK84] with respect to the generation of the envisionment graph. In addition, as each new node is added to the envisionment graph the actual behaviour of the devices is determined and checked against their purpose. If no conflict is detected the next node is generated until all nodes have been generated (*eg* see Figure 11). If a conflict is discovered, the envisionment process is halted and Doris identifies, as closely as possible, where in the system the conflict originates. The following two subsections describe these tasks.

4.1 Envisionment

The envisionment process has two distinct phases: the time during which devices change state and the time during which variables change state. At each node the new variable values are generated first and then the devices allowed to change state. Obviously in the real world these two types of changes occur simultaneously. However the limitations of a sequential evaluator impose certain restrictions on the way the real world can be modeled, therefore a notion of sequence has been imposed. Given that the time taken for variables to change is usually long compared to the time taken for devices to change it is not unreasonable to make this distinction. DeKleer and Brown [deK84] introduced the terms *normal time* and *mythical time* to describe these different time periods.

During normal time the system generates new nodes. This is done by allowing a single variable (chosen from the list of variables which are in a position to change) to change value. If n variables potentially will change at a given node then there will be a branching factor of n at that node. As discussed earlier, variables change value according to the logic set out in their definition. Once a new node has been created mythical time is invoked and the devices change state.

Those devices which use (in their control logic) the most recently changed variable are updated and if their behaviour changes then those devices to whom they are connected and those systems of which they form a part, are also updated. Mythical time continues until equilibrium is reached. This is essentially a bottom up process, from components to systems. However as the status of output ports change so the behaviour of connected devices on the same level of the hierarchy, also change. If these are systems devices then these changes are propagated down to the system's parts.

4.2 Purpose Analysis

The knowledge contained within the purpose description details certain behaviour patterns which must hold true if the device is to achieve its purpose. Therefore at every node in the envisionment graph this logic must be true. Purpose is described in the same terms as are used to define the status of devices and variables. For example, `'keep'` and `'active'` used in Figure 7 are defined using the language introduced in Section 2.

A number of approaches could be used to check that there is no conflict between the behaviour of a system and its purpose. Rather than checking every device at every node in the graph, the default method is to check that the purpose of the highest level device is satisfied. If there is no conflict at this level its parts are not examined since the assumption can be made that they are also achieving their purposes. This feature has been introduced for reasons of computational economy, but at the same time it is our opinion that this is a natural problem solving strategy.

It is possible to force a more exhaustive examination of the device and its parts by specifying which additional devices should also be checked. In systems which have built in redundancy, knowing that parts of the system may be failing even though the main system appears to be performing satisfactorily is an important consideration when analysing device behaviour. Of particular interest in such systems is the robustness of the overall system when failures occur.

If a conflict is detected, the reaction is always the same. The device hierarchy is unpacked, and an attempt is made to identify as closely as possible the source of the conflict (see Figure 12). If the conflict is associated with a system device (rather than a component) the parts of the system are each analysed. This process recurses down the hierarchy until either the problem is associated with a component or until no further conflicts are found. If no further conflicts are found the problem is at the systems level, *ie* the parts do not combine to achieve the desired behaviour through some fault in the engineering.

4.3 Example - The Greenhouse System

Figure 11 shows the envisionment graph produced for the greenhouse system. All nodes have been generated and no conflicts were found. The order in which the nodes are generated does not affect the shape of the graph. Doris generates the envisionment graph depth first which reflects implementation issues rather than any envisionment principle. The inner cycle represents the behaviour when the `'timer = on'`, `'temperature'` cycles between `'low'` and `'normal'`. The outer cycle represents the behaviour of the greenhouse system when `'time = off'`, `'temperature'` converges on `'low'`. The behaviour of `'v_clock, time'` is influenced by factors outside the greenhouse system and thus at every node it may change value, thus the bidirectional links between the inner and outer cycles.

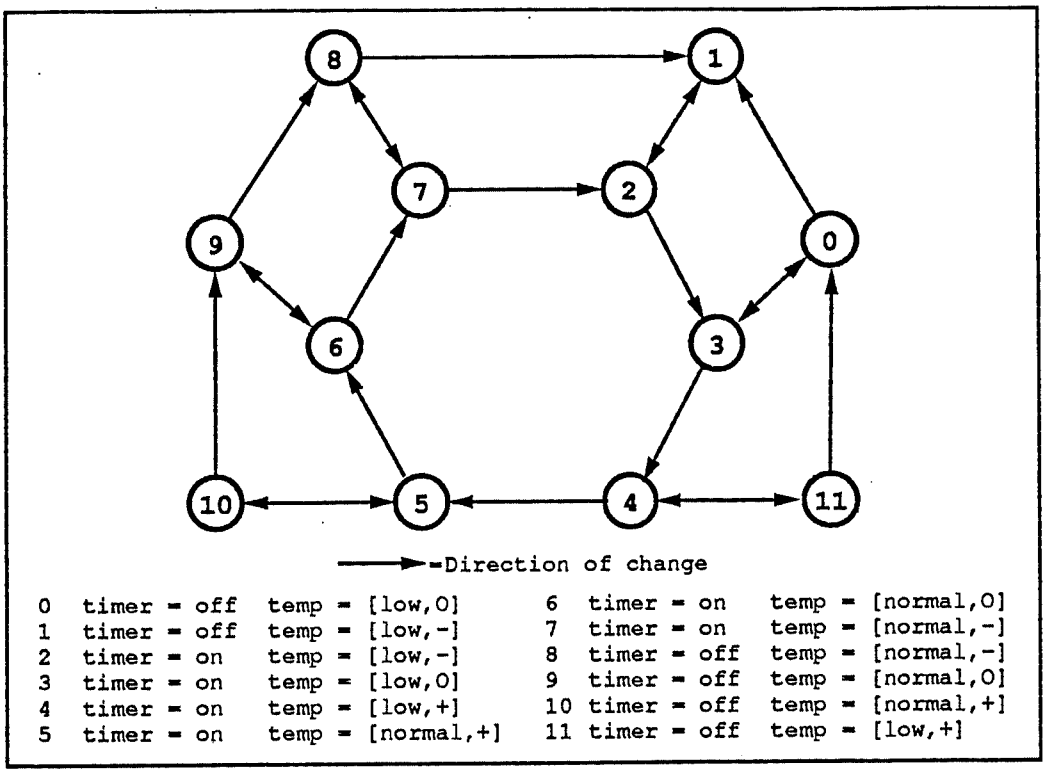


Figure 11: Greenhouse system envisionment

4.3.1 Scenario 1

Figure 12 illustrates the reaction when a conflict is detected. First the conflict is reported at the 'd_greenhouse_system' level. The purposes of its parts are examined and a conflict is found with the 'd_environ_control_system'. Its parts in turn are examined and the 'd_greenhouse-_thermostat' is found not to be fulfilling its stated purpose (NB the problem with the thermostat is due in part to the description of its purpose, the real purpose is to output electricty when temerature is low, the effect is to keep temperature normal). In addition it is reported that the connection between the 'd_greenhouse_thermostat' and the 'd_greenhouse_heating' does not correspond with the 'd_environ_system' general engineering technique. The envisionment graph produced up to the point when the conflict was found is also presented.

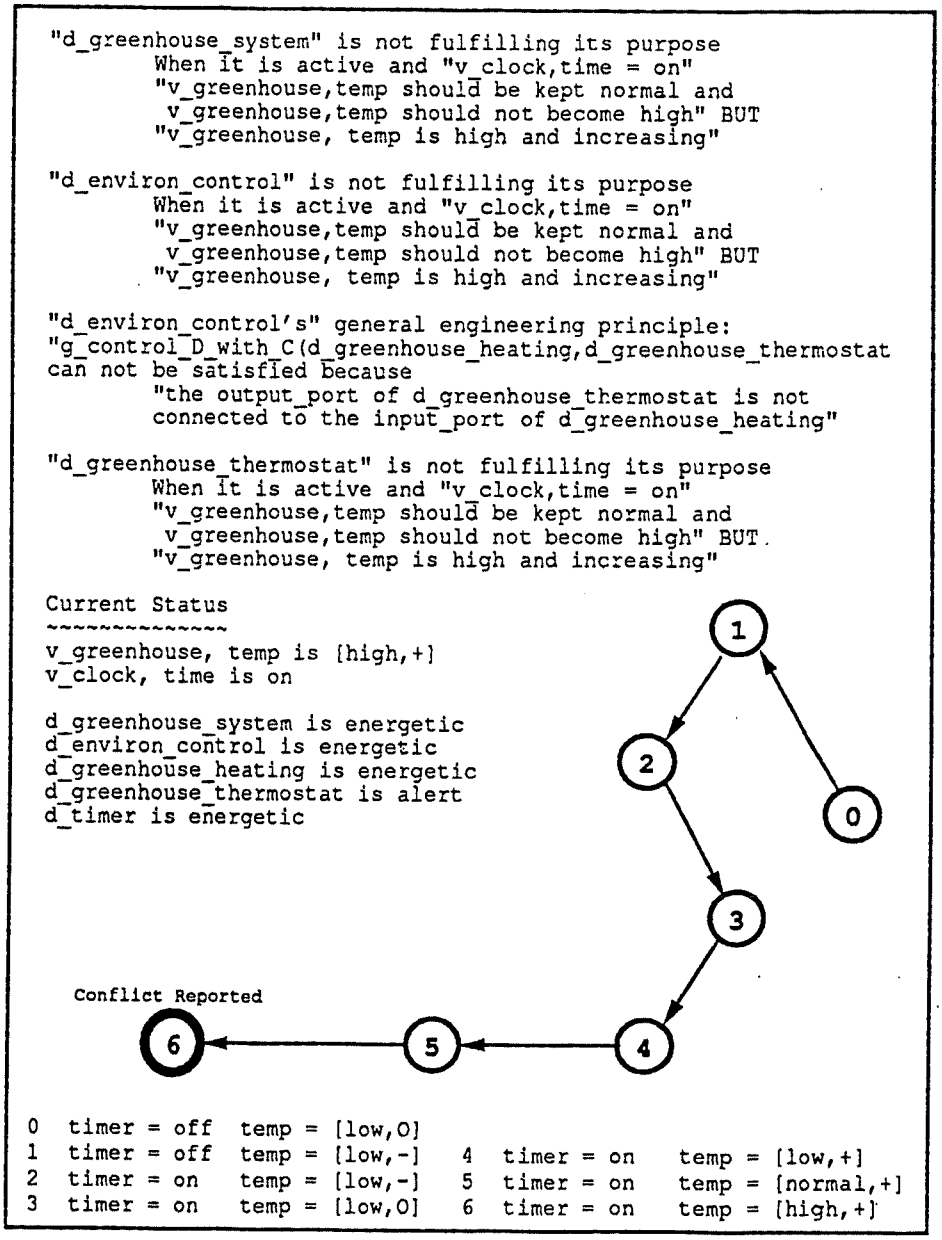


Figure 12: Doris output when thermostat unconnected

5 Other Applications

The representation described above has been developed with a view to providing a general representation of devices which can be used unaltered by other applications. Therefore no reference to, or knowledge of the design evaluation task is made within the representation. Furthermore a serious effort has been devoted to maintaining a clear *device centered* approach, as well as maintaining a clear separation between *Knowledge of Purpose* and *Knowledge of Structure*. Other applications for which this representation is applicable are simulation, fault diagnosis and explanation.

5.1 Simulation

Simulation as defined here is the task of forming the envisionment graph, not the usual task which maps the behaviour of a device in distinct time units along a straight line. The envisionment graph describes the whole set of the device's behaviour states and as such it gives a full and complete description of the device's behaviour. Envisionment not only permits the exploration of the behaviour of correctly functioning devices but also provides the means to determine the consequences of induced faulty behaviour.

It is important that designers understand not only the correct behaviour of their devices but also what behaviour would result given some failure within the device. This is especially true for devices with built in redundancy where it is important to understand what consequences failures within the device have on its overall behaviour. In these cases it is useful to understand the relationship between redundancy and robustness. On safety critical systems this is especially true, *eg* in aircraft hydraulic systems and nuclear power station cooling systems. Generating the envisionment graph for incorrectly working devices might therefore be as important as generating the graph for correctly working devices as the graph provides a basis for understanding the behaviour of the device even when it is faulty.

The problem illustrated earlier in Figure 12 was generated by introducing a fault into the knowledge base, the connection between thermostat and heater was broken. The envisionment process was halted as soon as the conflict was found. If purpose analysis is not required then it can be suppressed and a complete envisionment graph will then be generated, whatever the behaviour of the device.

5.2 Fault Diagnosis

Fault diagnosis can be thought of as the other side of the design evaluation coin in that the task is to identify which parts of a complex device are the source of unsatisfactory behaviour. During design evaluation the task assumes that the components behave correctly (unless it is explicitly stated otherwise) and it is their structural combination (the engineering) which is the focus of attention. In fault diagnosis the intention is to provide some indication of the nature and locality of the fault (thus focusing on devices) from a description of the external behaviour of the device. In order to perform fault diagnosis it must be assumed that the model correctly represents the system and that the model during design evaluation generates correct behaviour. Although the emphasis is different during fault diagnosis and design evaluation the problem solving mechanics are the same.

Our approach, as implemented in Doris, is to fix the variables to the observed values and then to perform the evaluation process as described above. Any conflict is identified and reported in just the same manner as above. Thus variable values are fixed to the values observed and an envisionment graph is generated. If all variable values are specified then the graph will consist of a single node. If a partial set of the values is provided then a partial graph is produced and the behaviour of each node is examined. The process of design evaluation is equivalent to fault diagnosis where no observed values are given and so a full envisionment graph is generated.

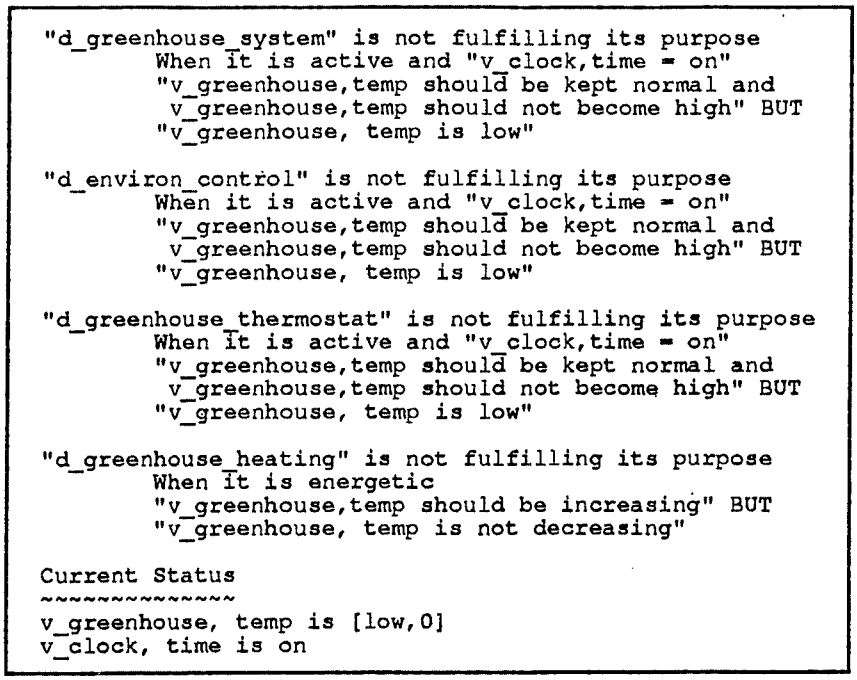
An interesting feature of this approach is that only a description of correct behaviour is used. The process of fault diagnosis describes how and why the observed behaviour deviates from the known correct behaviour by identifying the points of conflict within the system structure. This reflects our view that what is most significant about faulty behaviour is way in which it deviates from correct behaviour.

5.2.1 Example - Fault Diagnosis

Symptom: Greenhouse is cold when it should be warm (*ie* '`v_greenhouse, temp`' is low and '`v_clock, time`' is on).
Reaction from Doris: see Figure 13.

5.3 Explanation

There are two types of explanation: First, explanation which details the nature of the behaviour of the device and second explanation which describes the structure of the envisionment graph.



In this example Doris reports that there is a problem with the greenhouse system ('d_greenhouse-_system'). It tells us that it is the environment control system ('d_environ_control') part which yields the conflict. Within the environment control system both the thermostat and the heater are not behaving correctly. The reason for this problem is that the heater is not raising the greenhouse temperature ('v_greenhouse, temp'). (Note that the conflict associated with the thermostat is more a reflection of the description, which is not strictly true, than with the behaviour of the device. The thermostat's real, and more immediate purpose is to act as a switch and to allow electricity to flow when the temperature is too low, rather than to control temperature.)

Figure 13: Greenhouse too cold

First, if at the end of the envisionment process no conflict has been detected then the behaviour of the device can be explained in terms of the purpose it achieves, which is set out in the purpose definitions in the knowledge base. If more detail is required then the device's parts can be subjected to the same treatment.

Of more interest is an analysis and explanation of the envisionment graph. This project has not fully developed this application. However, there are certain features of the representation which provide a handle on the problem. Understanding the time relationship between variables (*ie* knowing how fast different variables change value with respect to each other) allows a partitioning of the graph into regions. The graph can also be divided into regions where devices maintain a particular status. These regions will, to some extent, represent the same parts of the graph, *eg* in Figure 11 the outer ring contains those nodes when 'v_clock, time' is off as well as when the timer is quiet. Given this ability to simplify a complex graph the task remains to describe the behaviour of the device within these sub-graphs, as well as the transition between regions.

6 Concluding Remarks

The main emphasis of this work has been on developing a representation strategy which handles Knowledge of Structure and Knowledge of Purpose. The crux of the matter, as discussed in [Cha86, Sem86 and Keu89], is to understand that Knowledge of Structure alone does not allow for a complete understanding of devices.

The articulation of purpose is the essential quality which groups components together to form systems. The ability to represent purpose allows for the creation of a hierarchy of systems and components. This nicely complements the consolidation work of Bylander [Byl85, Byl88] since we provide a clear basis for grouping devices into systems.

The representation of purpose provides constraints which reduce the 'frame problem' since it specifies which behavioral abstractions constitute interesting aspects of the device's behaviour. The terms used to describe purpose are defined using the language which describes device and variable status and this provides the mechanism for making the link between device behaviour and purpose description.

The development of a single representation strategy which represents both systems and components equally effectively allows different applications to traverse this hierarchy of devices easily and so allows them (and their designers and users) to concentrate on the task and not on the manipulation of the knowledge base.

Doris provides a meeting point between the work done by the Qualitative Reasoning community [deK84, For84, Kui84] and that being pursued by those involved with the development of the Functional Representation of devices [Cha86, Sem86, Keu89, Goe89]. This is a logical development since both are dedicated towards developing better techniques for representing and understanding the relationship between structure and behaviour.

Acknowledgements

We thank Helen Purchase and Julia Galliers for useful comments and discussions concerning this paper in particular, and this work in general.

References

[Byl88] Bylander, T., Smith, J. and Svirbely, J., Qualitative Representation of Behaviour in the Medical Domain, Computers and Biomedical Research, 21, pp 367 - 380, 1988.

[Byl85] Bylander T. and Chandrasekaran, B., Understanding Behaviour using Consolidation, in Proc. 9th. International Joint Conf. on Artificial Intelligence, Los Angeles, 1985.

[Cha86] Chandrasekaran, B., Josephson, J. and Keuneke, A., Functional Representation as a basis for explanation Generation, in Proc. IEEE Int. Conf. on Systems, Man and Cybernetics, pp 726 - 731, 1986.

[deK84] de Kleer, J. and Brown, J., A Qualitative Physics based on Confluences, Artificial Intelligence, 24 (special issue ed. D. Bobrow), pp.7 - 84, 1984.

[For84] Forbus, K., Qualitative Process Theory, Artificial Intelligence, 24 (special issue ed. D. Bobrow), pp.85 - 168, 1984.

[Goe89] Goel, A., Integration of cased based reasoning and model based reasoning for adaptive design problem solving, PhD thesis, Ohio State University, 1989.

[Keu89] Keuneke, A., Machine understanding of devices causal understanding of Diagnostic conclusions, PhD Thesis, Ohio State University, 1989.

[Kui84] Kuipers, B., Common Sense Reasoning about Causality: Deriving Behaviour from Structure, Artificial Intelligence, 24 (special issue ed. D. Bobrow), pp 169 - 203, 1984.

[Kui86] Kuipers, B., Qualitative Simulation, Artificial Intelligence, 29, pp 289 - 338, 1986.

[Sem86] Sembugamoorthy, V. and Chandrasekaran, B., Functional Representation of Devices and Compilation of Diagnostic Problem Solving Systems, in Experience, Memory and Reasoning (eds. J. Kolodner and C. Riesbeck), pp 47 - 73, 1986.

John Bradshaw completed his MSc in 1986. He has been a research student at the University of Cambridge since 1987 and is now in the final stages of his PhD.

Richard M. Young studied Engineering and Artificial Intelligence (AI) before gaining his PhD in Psychology from Carnegie-Mellon University in 1973. From 1973 to 1978 he was a Research Fellow in the Department of AI at Edinburgh University. Since 1978 he has been on the scientific staff of the UK Medical Research Council's Applied Psychology Unit in Cambridge. He also acts as a consultant to Rank Xerox Cambridge EuroPARC. His research interests lie in human-computer interaction and the use of AI for the computer simulation of human thinking, problem solving and cognitive skills.