Destructive Problem Solving – a Novel Approach to Configuring

Werner E. Juengst

Freightliner Corporation
4747 North Channel Avenue
Portland, Oregon 97217
WernerJuengst@Freightliner.com

Abstract

The paper discloses a novel method for configuring a technical system through model-based reasoning from a description of its variant spectrum in a master circuit diagram (patent pending).

After describing background and basic idea, the basic method and a generalization are explained. Knowledge representation and reasoning process are illustrated in two examples.

Introduction

In many engineering domains, e.g. electrical, electronic, hydraulic or pneumatic power and control systems, the prime representation of engineering decisions is in the form of circuit diagrams. An engineer with a circuit diagram has no need of a knowledge base with explicit rules.

The ease and absolute certainty with which engineers can draw conclusions from such diagrams made and makes it very tempting to employ such graphical representations for the knowledge and for the results of the configuring process. When circuit diagrams are available, why should our computer-based configuration tools still have need of separately maintained rules?

Motivation

Manufacturers of complex products have no real choice but use some form of knowledge representation for describing their product offering and determining the configuration for their products. So they employ the existing inefficient rule-based technologies in spite of the large maintenance effort and accept the huge price tag and the shortcomings.

Examining more closely what manufacturers are willing to undergo today, some preconceptions of what model-based techniques should strive for were found to be astray. Most important is that manufacturers are prepared to describe the product feature combinations they want to offer in rules of propositional logic, i.e. completely, extensively and explicitly, whereas the preconception of researchers is to treat configuration space as unlimited, and construct the solution incrementally from a minimized knowledge base.

Another observation is that many manufacturers today create and maintain, for every component (or for every partial bill-of-material) that might be put into the product, its own rule in terms of the propositions in the product specification.

So from a manufacturer's viewpoint there is no reservation about how a circuit diagram might be employed as long as it reduces the burden of creating and maintaining the knowledge bases for configuring.

Background

Rule-Based Approaches. Early attempts to configure with circuit diagrams [1] stayed close to the tradition of rule-based expert systems. The possible circuit diagrams were dissected into tiles with partial diagrams corresponding to functional groups of elements. The partial diagrams were carefully arranged so that tiles could be put together like in a puzzle to form various circuit diagrams, and would span the whole configuration space by the possible combinations of alternative partial diagrams. Which tiles, i.e. partial diagrams, were necessary for a given requirement specification was determined by rules. The position of each partial circuit diagram was decided beforehand.

This technique was viable and successfully and profitably employed in some applications. However, the effort required for knowledge base maintenance was very high. In addition to maintaining the rule base that determined the presence of the partial diagrams, and separate from it, it was necessary to design the partial circuit elements while making sure that they fit together and formed legal and functional total circuit diagrams.

Model-Based Approaches. While knowledge representation by circuit diagrams led to breakthrough successes in model-based diagnosis systems [2], developers of model-based techniques for configuring do not consider circuit diagrams promising. For incremental construction of configurations, a circuit diagram offers too many alternatives of adding circuit elements to have attractive combinatorial characteristics. Also, the function of a circuit is much more strongly determined by the connectivity structure of the components than by their type, number and parameters. Thus, the intended function of a circuit does not arise until most of the components are present and suitably connected, and incremental configuring is left without local criteria for focussing on promising evolution directions, a difficulty that became the show-stopper for the method of „Constructive Problem Solving" [3]

In model-based reasoning for configuring, e.g. in the resource-based paradigm [4], the basic principles of engineering for technical systems are applied: the principles of Necessity and of Minimality. Necessity postulates that no component should be in a technical system unless it is necessary for the creation or operation of the system. Minimality postulates that in the choice between alternatives, the alternative with the minimal global price tag should be selected. Can we apply this kind of reasoning with a circuit diagram?

Origins of the Idea

In many application areas, products are manufactured with more components than are needed in every application case. These products can be „configured" by means of circuit bridges that enable the necessary and disable the unnecessary optional subcircuits. The circuit bridges are set properly before the circuit is used for the first time. In the circuit diagram, these bridges are identifiable components, and their connections are designed such that the subcircuits are properly enabled or disabled depending on the state of the bridge.

In each case, the disabled components, though physically present, are without function and therefore superfluous. If we wanted, we could remove those components from the product without affecting its function as configured. It was from this intuitive picture of starting with an all-encompassing configurable product and then ripping out and throwing away the unnecessary components in order to arrive at the configured product that the name „Destructive Problem Solving" suggested itself.

As we see, the circuit diagram can be and is used to describe products that can be configured. Moreover, if the manufacturers would know beforehand into which state the configuring switches and bridges will be set in each case, they could determine from the circuit diagram what subcircuits will be disabled. They then could leave out of the product all components of those subcircuits, i.e. use the circuit diagram and model-based reasoning to determine the configuration of their product.

Method

What we need is a model-based reasoning mechanism that can distinguish between functional and functionless components and hence necessary and unnecessary components in a circuit diagram.

Basic method

Circuit diagrams show components and their connections. If the components are to work properly and a subcircuit is to deliver the expected functionality, the components and the connections shown in the circuit diagram for the subcircuit are necessary. So, if a component represented in the circuit diagram of a configurable circuit can be proven to be necessary, all circuit elements that are connected to that component must be considered necessary unless proven otherwise. On the other hand, if reasoning about the connections of a component leads to the conclusion that the component is without function, then it may (from a functional view) and must (from the principle of Necessity) be removed from the circuit diagram of the configured product. After the removal, because of the connections left open by the removed component, there usually are other components for which it becomes obvious that they do not contribute any functionality to the technical system, and that they too may and must be removed.

Typically, a component or subcomponent that is unconnected at one terminal can immediately be recognized as being obviously without function, e.g. a wire, a motor, a relay coil or a lamp.

Components with remaining functionality. For some types of components and some connections, leaving one connection open does not take away the functionality completely. In such a case, we must replace the component represented in the circuit diagram by a type of component that exhibits the remaining functionality. This new component may again leave some of the remaining connections open, just like when a component is removed.

In the case of a relay, a disconnected coil will leave the relay permanently de-energized, so that the state of the contacts shown in the circuit diagram (by convention the position in the de-energized state) is permanent. The contacts of the relay can then be replaced by permanent connections (normally closed contacts) or removed completely (normally open contacts) and the resulting changes in connectivity used for further reasoning. In other cases it may instead be obvious that a relay coil is permanently energized, so that we can infer a stable state of the relay contacts opposite to that depicted in the circuit diagram.

Multiple-state components. That line of reasoning can be extended to components that can have more than one state and in each state effect only one of a number of possible connections, e.g. a switch. If it can be inferred that such a component needs to be permanently in a certain state for the proper working of a necessary component, then all other connections are proven to be permanently open. This, again, may make it obvious that some components cannot contribute to the overall functionality and thus may and must be removed.

Components with subcomponents. In other cases, it may be that only some subcomponents of a component will be proven to be superfluous. The principle of Minimality then demands that the component type with the least, but still sufficient, overall functionality should be substituted. The resource-based paradigm has proven to be a very efficient technique for this kind of reasoning.

A typical example is a relay with multiple contact sets. If some of the contacts are proven unnecessary, only these contacts may and must be removed. The other contacts and the relay coil may still be necessary.

However, if all of the contact sets of a relay are proven to be unnecessary, then obviously the relay coil itself has no more reason for being, and can be removed as well.

Knowledge representation.

How must we represent the knowledge for „Destructive Problem Solving"? We define the configuration space by means of a master circuit diagram, as if we wanted to construct and manufacture one single product implementation from which all intended configurations can be obtained by the setting of bridges. The master circuit diagram is maximal in that it comprises all circuit elements and all connections that are needed under some circumstance. We then must assign to each of the configuring bridges the conditions that determine its proper state from the attributes of the requirement specification that the customer will submit. This represents the configuration knowledge base. See Fig. 1 for an example.

Reasoning Process

The reasoning process in Destructive Problem Solving begins when the requirement specification for a given case is complete (enough). We first evaluate the conditions that determine the state of bridges for that case (Fig. 2).

Beginning with these bridges, we iterate through all components that were connected to unnecessary components to find out whether they are necessary or unnecessary or whether their internal state is permanently determined, and remove components that can be determined to be superfluous (Fig. 3). The iteration naturally ends when we have established for all components in the circuit diagram whether they are necessary or unnecessary (Fig. 4).

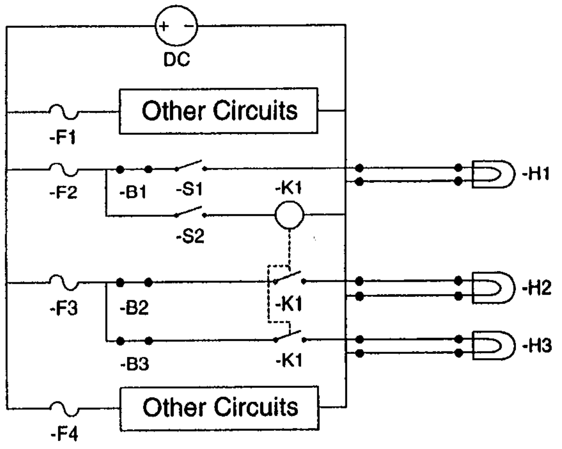
Example

For the example let us consider the „Master Circuit Diagram" and the bridge conditions shown in Fig. 1.

Besides the power supply and other circuits, it shows four subcircuits that control three lamps. The fuse -F2 is common to two of them. The first subcircuit controls lamp -H1 and consists of bridge -B1, switch -S1, lamp -H1 and suitable wires. The second subcircuit consists of switch -S2 and the relay coil of relay -K1. Fuse -F3 is shared by the other two subcircuits. Each consists of a bridge (-B2, -B3), a contact of relay -K1, and a lamp (-H2, -H3). The interior light -H1 is activated directly by switch -S1. The exterior lights (-H2, -H3) are controlled jointly by switch -S2 through relay -K1. Such a construction is usual when lamps draw high currents or operate at a voltage different from the control circuit.

The circuit is configured by setting the states of the bridges. In the requirement specification, the customer can decide separately about each light. The conditions for the states of the bridges, which link the circuit diagram to the requirement specification, are here shown in a separate list besides the Master Circuit Diagram, but will best be associated with the circuit elements they refer to.

Application Case. In the first application the customer required „Interior Light" and „Rear Light". From the conditions in Fig. 2 we infer that the state of -B1 and of -B3 must be CLOSED, the state of -B2 must be OPEN.



COMPONENT STATE CONDITIONS

-B1:
REQUIRED("Interior Light") : CLOSED
ELSE : OPEN

-B2:
REQUIRED ("Front Light") : CLOSED
ELSE : OPEN

-B3:
REQUIRED ("Rear Light") : CLOSED
ELSE : OPEN

Fig. 1 Example of a Master Circuit Diagram together with the bridge conditions

SPECIFICATION
...
Interior Light Required.
Rear Light Required.
...

COMPONENT STATES

-B1: CLOSED
-B2: OPEN
-B3: CLOSED

Fig. 2. Specification and bridge states for case 1

Because bridge -B2 is open, the wire leading from -B2 to the contact of -K1 will never conduct current, it is therefore proven superfluous for this case. The same then is true for the contact of -K1, for the wire leading from it to lamp -H2, for the lamp -H2, and for the wire leading back from -H2 to the Minus crowbar. These components, shown with gray background in Fig. 3, therefore can be removed from the circuit diagram and need not be put into the product.

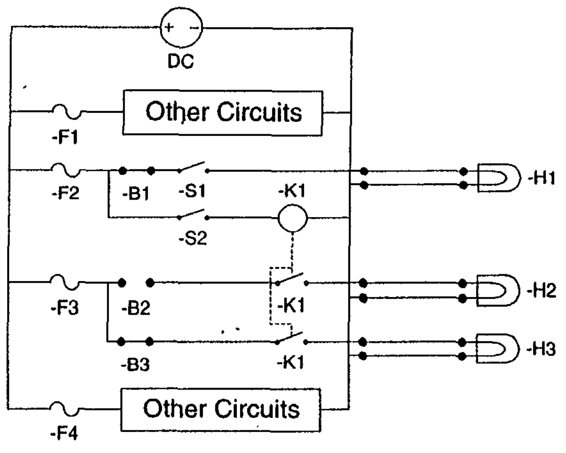


Fig. 3 Intermediate state of circuit diagram with circuit elements proven unnecessary (case 1)

As bridges -B1 and -B3 are permanently closed, they may be merged with the wiring. This leads to the circuit diagram in Fig. 4 for the configured product.

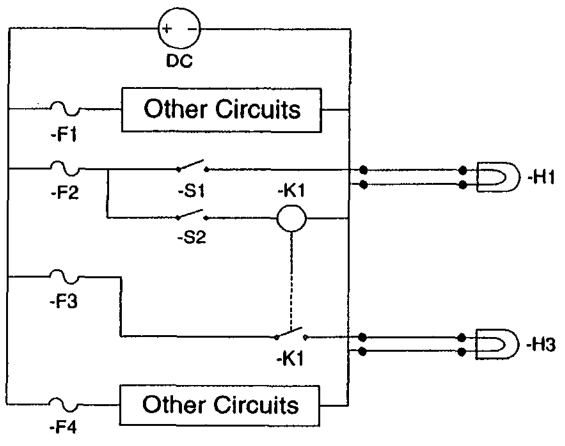


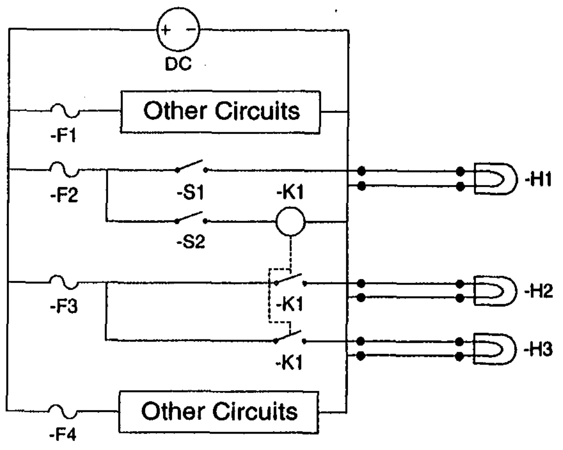
Fig. 4 Final circuit diagram of the configured product (case 1)

Because only one contact of the relay -K1 remains in the circuit diagram, the relay -K1 can be implemented by any type that provides at least one contact of sufficient current rating.

Generalizations

The inferential coupling between the attributes of the requirement specification and elements of the circuit diagram need not be limited to bridges. In many cases, the conditions can be directly attached to an functional element of the circuit diagram if attributes of the requirement specification relate to the existence or absence of that functional element. These conditions are then evaluated together with the bridge conditions, the state of the components are determined, and all "open" components removed at the start.

That will typically be suitable when the customer can directly decide about the presence of elements, e.g. sensors, gauges, indicators or other key functional elements.



```
COMPONENT STATE CONDITIONS

-H1:
REQUIRED( "Interior Light" ) : LAMP
ELSE : OPEN

-H2:
REQUIRED ( "Front Light" ) : LAMP
ELSE : OPEN

-H3:
REQUIRED ( "Rear Light" ) : LAMP
ELSE : OPEN
```

Fig. 5 Example of a Master Circuit Diagram without bridges together with the component state conditions

Example

Fig. 5 shows the same example as Fig. 1, but with conditions attached to functional components and not to bridges.

Application Case. For our second case, let us assume the customer required only the interior light. Fig. 6 shows the resulting states of the components.

```
SPECIFICATION
...
Interior Light required.
...
```

```
COMPONENT STATES

-H1: LAMP
-H2: OPEN
-H3: OPEN
```

Fig. 6 Specification and component states in case 2

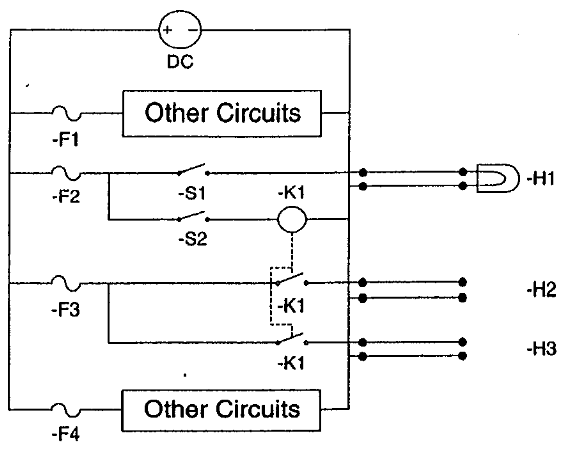


Fig. 7 Intermediate state of circuit diagram with circuit elements proven superfluous first

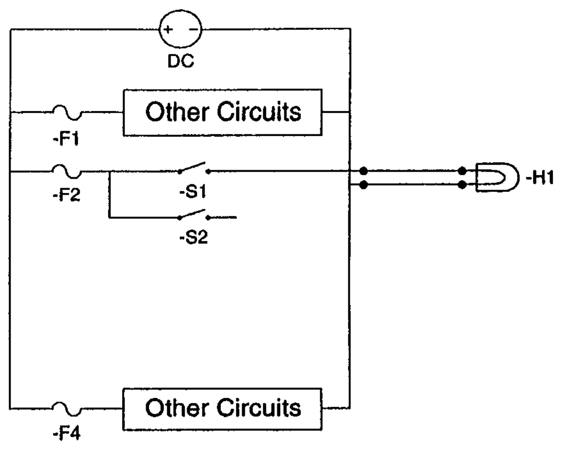


Fig. 9 Intermediate state of circuit diagram with circuit elements proven superfluous finally

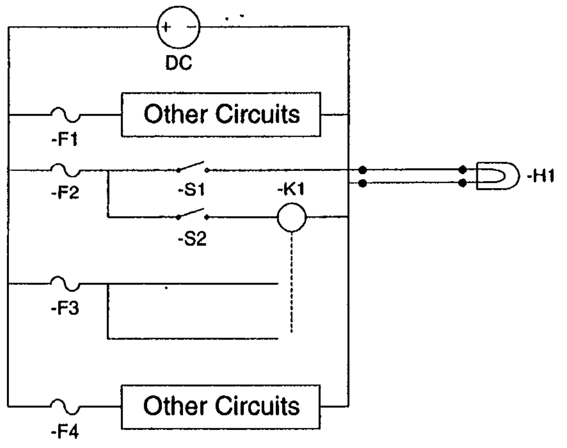


Fig. 8 Intermediate state of circuit diagram with circuit elements proven superfluous next

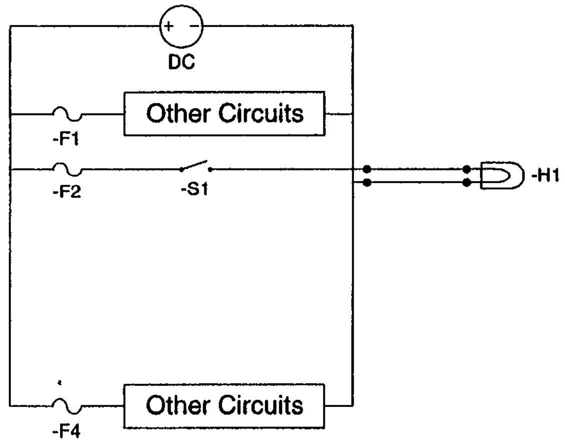


Fig. 10 Final circuit diagram of the configured product in case 2

In a first step, both -H2 and -H3 were removed by the inference about the component state. The related circuitry, i.e. the wire leading to the lamps, and subsequently the relay contacts, are unconnected at one terminal (Fig. 7). Thus they are superfluous and can be removed from the configuration.

After removing these (Fig. 8), we see two open-ended wires that must be removed. This leaves one terminal of fuse -F3 unconnected to any circuitry, so that no current can flow and -F3 is proven superfluous. Also, the relay coil of relay -K1 is not connected to any contact, so it is without function and proven superfluous. Both can be removed.

But then, in Fig. 9, it now is obvious that switch -S2, because one of its terminals is unconnected, can never conduct current. So it is superfluous and must not be present in the final configuration.

After these removal operations we arrive at the final configuration (Fig. 10).

Further Generalizations

Multiple alternative states. A further generalization is to allow more than one alternative state for a symbol under different conditions.

This can be used e.g. when, depending on the voltage, an indicator light can be alternatively implemented as a lamp, an LED, or a neon light.

For some circuit elements the replacement symbols representing them under different conditions may even have different connectivity, e.g. when a switch has different numbers of switch positions under different conditions, or when an electrical motor has or does not have taps for coil-switching dual-speed operation.

In each of those cases, the state condition specification for the component will include more alternative states.

More expressive conditions. In principle, complex conditions for the existence of subcircuits can be represented by suitable networks of bridges. Allowing logical expressions of predicates in formulating the condition for a component state, and providing other predicates on the content of the requirements specification than "REQUIRED", e.g. comparisons, enhances the expressive power and conciseness of the knowledge representation without increasing the complexity of the model-based reasoning process itself.

Summary

Circuit Diagrams are suitable to represent configuration knowledge in an extensive fashion. The configuring process in Destructive Problem Solving relies on the fact that only necessary circuit elements need to be implemented in the product, all others can be removed from the circuit diagram and left out of the product's configuration. The representation allows model-based reasoning about circuit elements to determine whether circuit elements are necessary or unnecessary from the existence of their connections with other circuit elements in the circuit diagram.

The link between customer specifications and the circuit diagrams is achieved through describing state conditions for key circuit elements in the master circuit diagram.

Expressing the configuration space of a product by means of a circuit diagram is a task familiar to electrical engineers and technicians, a very much larger group of people than that of knowledge engineers educated in expressing the configuration space through logical rules or constraints. Thus, manufacturers can draw on a much larger labor pool and in most cases already have people of such qualification in their work teams.

References

[1] Heinrich, M. 1989. Expert System for the Preparation of Circuit Diagrams, *Automatisierungstechnische Praxis atp*, 31(4):190-195.

[2] Tatar, M. 1995. Diagnosis with Cascading Defects. Proc. DX'95, Sixth Int. Workshop on Principles of Diagnosis, 1995, Goslar (Germany).

[3] Klein, R.; Buchheit, M.; and Nutt, W. 1994. Configuration as Model Construction: The Constructive Problem Solving Approach. In: Gero, and Sudweeks eds. 1994. Artificial Intelligence in Design '94. Kluwer, Dortrecht, The Netherlands, 201-218.

[4] Juengst, W. E.; and Heinrich, M. 1998. Using Resource Balancing to Configure Modular Systems. *IEEE Intelligent Systems* 13(4):50-58.