Report No. 6560

OREO - ADDING ORIENTATION TO A DYNAMIC QUALITATIVE SIMULATION

Frank Ritter

August 1987

Prepared by:

BBN Laboratories Incorporated
10 Moulton Street
Cambridge, Massachusetts 02238

Table of Contents

List of Tables

Abstract

In order to simulate electrical circuits accurately using a qualitative model, it is necessary to give parts orientation with respect to the voltage source in the circuit. Beginners' mistakes such as not finding a short across a part can often be attributed to incompletely orienting the circuit. Experienced engineers successfully orient circuits, though on an implicit level; they usually don't mark such things down on paper. We explain an algorithm (OREO) to orient any circuit based on the way experienced engineers orient circuits. The algorithm is able to explain itself and recognize shorts, bridge elements, and other paths useful for a quantitative simulation. The algorithm allows qualitative systems to be dynamically altered and is upwardly compatible with numeric simulations. A detailed example is worked with figures showing the internal structures that OREO uses. A complete explanation of the algorithm is included as the Appendix.

1 Introduction

The circuit model that OREO (a program to ORient circuits) helps to simulate was chosen for its simplicity, as well as for its accuracy. The top level program, QUEST (Qualitative Understanding for Electrical Simulation and Training), is designed to teach basic circuit theory and troubleshooting. This qualitative model of electrical circuit theory (included faulted components) was developed by White and Frederiksen [White, Frederiksen 86a], [White, Frederiksen 86b], [White, Frederiksen 87]. The model was designed to be upwardly compatible with more detailed and more numerically oriented models; you shouldn't have to unlearn to progress.

QUEST provides a powerful but simple 0th order qualitative simulation that is both accurate, and teachable. It is easy for the program to explain circuit behavior in terms students can understand because internally QUEST uses the same series of qualitative models that are being taught to the students. Other systems may have to translate their electrical model's output into terms appropriate for the student (for example Sophie [Brown, Burton, Bell 74]). QUEST has the problem of choosing how completely it wishes to explain its internal mechanizations. It is simple to provide any level of explanation to students; choosing the appropriate level of detail is the question that is left.

One of the basic procedures that can be taught is path tracing, which is an essential tool for finding out if there is a voltage drop across part. This idea is part of the qualitative simulation of electrical circuits that engineers use, and is used in QUEST. Checking for a voltage drop breaks down into two questions: "Do you have a good path to ground?" and "Do you have a good feed path?" By knowing just about voltage drops, it is possible to predict simple circuit behavior, and one can troubleshoot simple faulted circuits with only a test light.

One of the major user modes of QUEST allows the student to follow along as the program runs the simulation. If the program wrongly traces a path, someone attempting to follow the program's reasoning is likely to get confused. In QUEST this would happen without OREO. When an engineer looks at a circuit, one of the most important things that he/she does is to orient the elements with respect to the power and ground by indicating the polarity of the voltage drop across the device. He/she does this by marking the side electrically closest to the power (usually a battery) with a plus (+), and the side electrically closer to ground with a minus (-). The plus (positive) side is electrically closer to the battery, and the minus is further away. This orientation may not be readily apparent. The graphic depiction of the circuit may imply the opposite polarity requiring some analysis rather than simple

inspection. Adding orientation facilitates later searches for paths through the circuit, and internalizes the circuit's topology for the engineer. Any cognitively based qualitative simulation must do this also. In addition to aiding the clarity of the explanations, orientation of the circuit dramatically speeds up the search process.

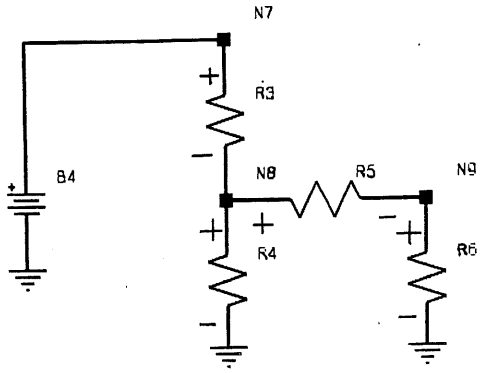
To troubleshoot a circuit in QUEST, the student can insert a test light, disconnect a part, or replace a part. These actions occur too often to store the orientations for all the possible circuits that can be constructed from the initial circuit. Dynamically computing the orientation of the circuit after one of these actions is a realistic problem that has to be solved by the simulation and by students.

Let us explore the situation of the unoriented circuit. One of the common mistakes that QUEST would make without OREO would be to incorrectly trace paths that did not have the default (as taught in schools and used in industry) clockwise orientation. This is a common mistake for beginners, while experts will often redraw the circuit to conform to this convention. Figure 1a shows the internal representation of a circuit.[1] Figure 1b depicts the circuit after a part has been added (wire W6), and also the incorrect internal representation that is the result of orienting only the new part with the default orientation. (For a wire this means to leave it unoriented because there is not a voltage drop across it.) Figure 1c shows the correct way to orient the circuit, done by QUEST with OREO: when a change is made to the circuit, the orientation of all circuit components is recomputed.

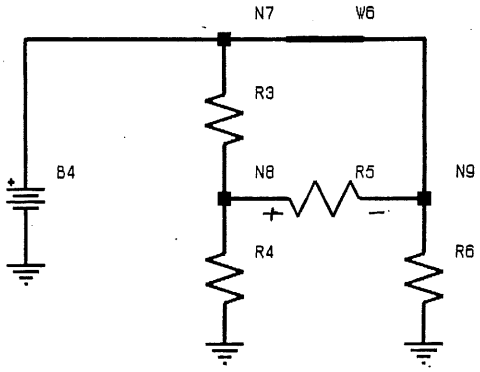
If we were using a quantitative analysis, with loop or node equations, then orientation would not matter. If an orientation was reversed, we would find a negative voltage, which we would invert along with the orientation to make the voltage positive. Reasoning about circuits in this way, updating initial assumptions, is an important type of analysis, but we do not want this analysis intruding on our explanations to novices. We are using a qualitative model because it provides more natural and monotonic explanations.

Mistakes in determining voltage drops will often result when path tracing is done without the orientation fixed to each part. Figure 1 illustrates a mistake that didn't influence whether there was voltage drop across R5, but did influence its direction. In Figure 2 we can see that an addition of a wire without correct reorientation could cause an incorrect voltage drop across BULB1. Before the addition of W1, BULB1 had a voltage drop across it. With the

[1]The dashed lines and orientation notation represent internal structure and are not currently displayed to the student. Available information occasionally may be withheld to make the figures more clear.

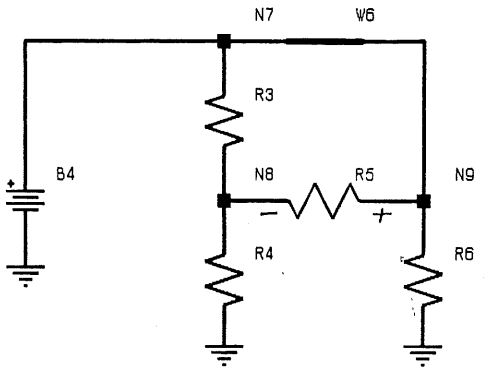


a) Original circuit



b) Addition of wire W6

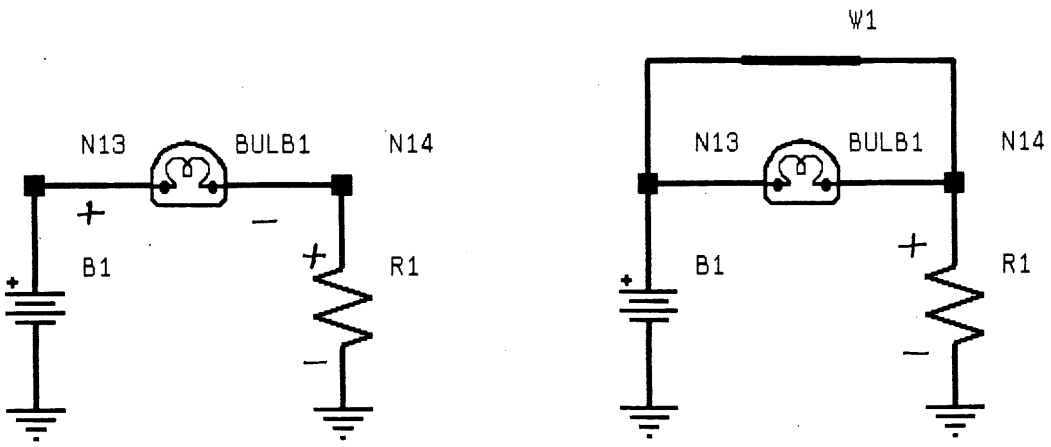
(incorrect orientation)



c) Correct orientation

1. Adding a part can change a part's orientation

wire attached however, there should not be a voltage drop. When one fails to reorient the circuit or check for shorts after the addition of a part, even for linear components, an accurate simulation is not guaranteed. QUEST originally checked for shorts, but this is inefficient, is not the way engineers view it, and is difficult to do completely and correctly.



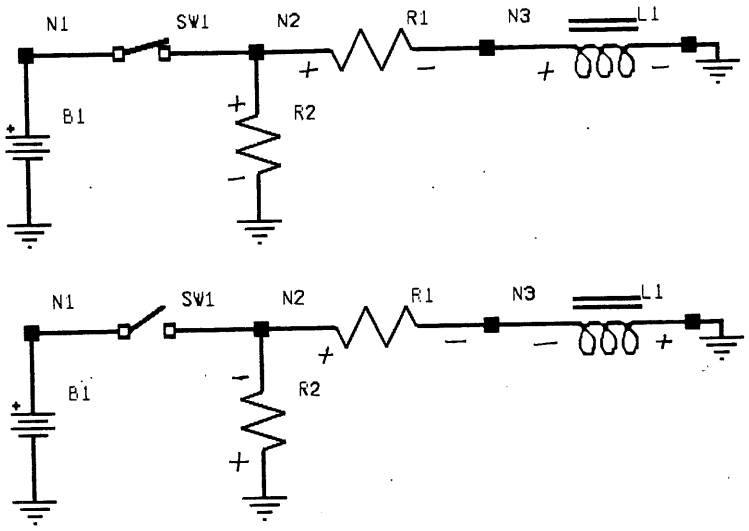
a) Original circuit

b) Addition of W1 removes voltage drop across Bulb1

2. Adding a part can remove a voltage drop

The addition of a wire is electrically equivalent to a switch closing or a capacitor becoming discharged. In electrical terms, a wire has no resistance, and thus no voltage drop across it. In constraint propagation, it indicates that the constraint should be completely and freely propagated. Thus inserting a new part, or changing a part's internal conductivity, is a sufficient condition for reorientation.

Figure 3 shows that one must also reorient when a part changes state because the voltage source, not just the paths, may change. In this example, an inductor starts to discharge and becomes a voltage source when it is disconnected from a battery.



a) B1 is the voltage source (top)

b) L1's field collapses making it the voltage source. Polarity changes on R2.

3. Changing a voltage source forces reorientation

2 Electrical Foundations

We have shown that in order to determine if there is a voltage drop across a device, the orientation of all the circuit components must be correct. We would also like to have a simple, easily learned method for the generation of the orientations. Thus, we have chosen to separate this orientation generation process from the basic voltage drop analysis.

Looking for shorts, and mixing the orienting and the tracing of paths is a potential source of confusion. Looking directly for a short across several parts is expensive computationally, and is a tricky business as we indicated in Figure 2. When electrical engineers analyze a circuit, they often use heuristics; sometimes they

recognize a configuration and use a "compiled" subcircuit for it. They will almost certainly not systematically apply the rules explained below. But beginners can not use heuristic rules or compiled knowledge. They lack the familiarity and knowledge structures of the mature analyst. Trying to follow an expert's apparently random walk through a circuit will appear arbitrary and difficult to understand. We propose a simple and predictable method that ensures understandability and teachability.

2.1 The Algorithm

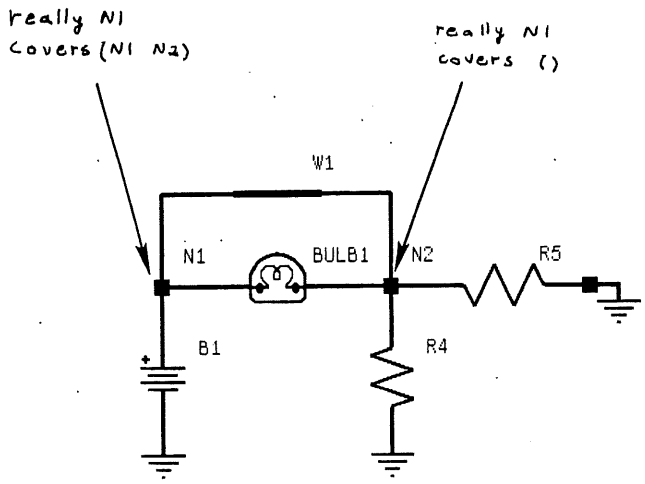
The algorithm we propose basically is: 1) to note the electrical nodes in the circuit by grouping the graphical nodes known to have the same electric potential together; 2) to reduce the circuit to a simpler circuit by compressing the series and parallel paths in the circuit to their simpler equivalent circuits; 3) when the circuit can no longer be compressed, to orient the paths that are left; and 4) to expand the circuit using the new orientation information and the information that was saved as the circuit was compacted. The details and reasons for these steps are explained in the following sections.

2.2 Avoiding Common Mistakes

The first step that we take in our analysis of a circuit is to create the distinction between graphical and electrical nodes (e-nodes in the Appendix). Graphic nodes are nodes that are used to mark the connection of two or more components. If two nodes are connected by a conductive-non-resistive path (for example a wire), they are both at the same voltage, and are electrically the same node. If these two graphical nodes are marked as being the same electrical node before analysis, one need not exhaustively search for shorts while carrying out further analysis. Thus OREO's first step is to find all the electrical nodes.

Looking for shorts is transformed into checking if the part is hooked up on both ends to the same electrical node. Figure 4 provides an example where BULB1 is hooked up on both ends to electrical node N1. Looking at nodes this way clarifies the problem by separates it into subtasks. One no longer has to check for shorts when tracing a feed path. Finding non-local shorts is a common problem for beginners.

To emphasize the pitfalls of always checking for shorts, consider the case of a web made up of shorts, Figure 5. Orientation of this group of wires does not make sense because they are all at the same potential. For our



4. A shorted part

purposes tracing a path through such a network is meaningless because the result is a multitude of paths with no resistance.

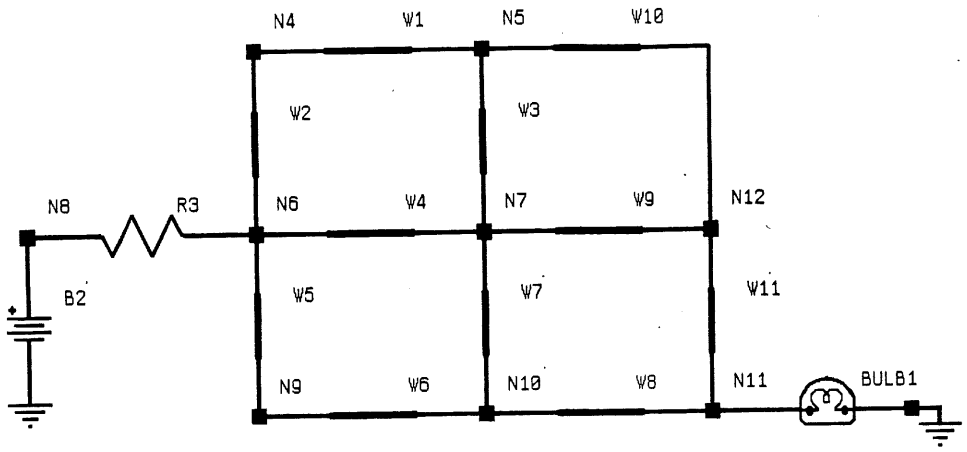
There is a lesson to be learned here concerning constraint propagation. If constraints are propagated through a network that may include elements analogous to wires (that is, objects that immediately propagate the full constraint, null elements), or if the network is not a simple tree or loop but a series-parallel graph where the depth of each node in the graph can change with respect to time, then the network should be compacted and oriented in a way analogous to OREO. Orienting the network this way allows the constraint propagation to proceed correctly even when there are (non-feedback) loops.

2.3 Compacting an Example Circuit

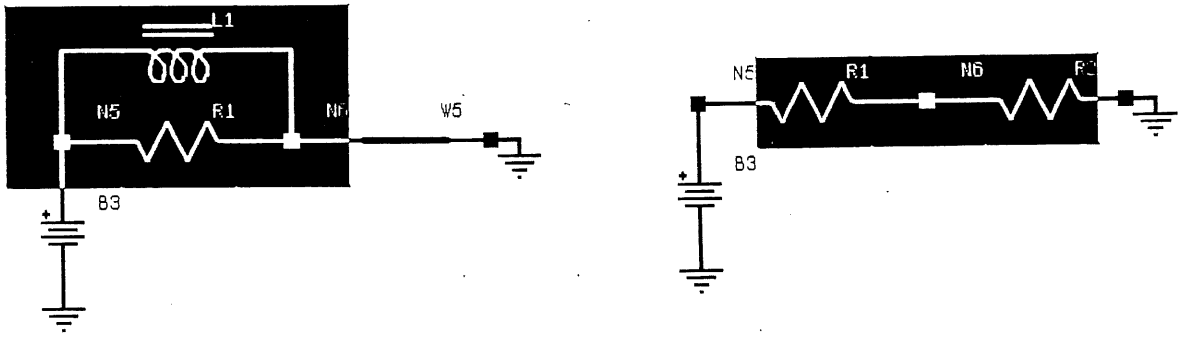
Now that we have only real electrical nodes, we can start to reduce the circuit. A circuit can be viewed as a collection of paths, where a path is a part, or a collection of parts and/or paths. The standard definitions of series and parallel paths are used. Figure 6a shows an example of a parallel path. Figure 6b shows a series path.

One of the simpler "tools" taught a beginning electrical engineer is the notion of equivalent circuits[2]. In brief,

[2]Ch. 5 of Trick's book [Trick 77] has a much more detailed explanation of how this works.



5. A web of shorts



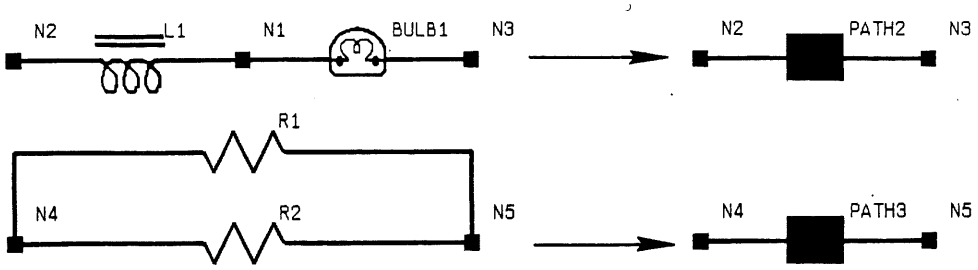
a) Series path

b) Parallel path

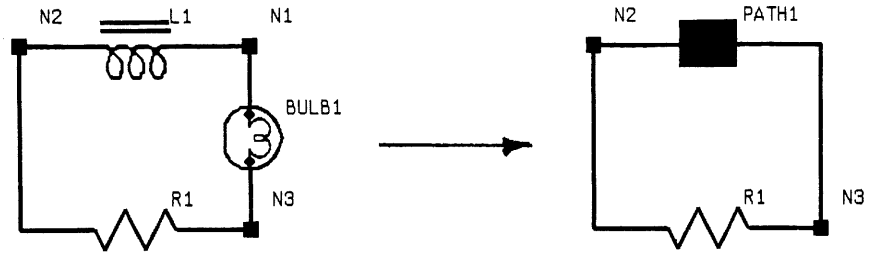
6. Examples of series and parallel paths

this works as follows: series and parallel circuits are replaced by their equivalent series or parallel path. This reduces the circuit, until there are no more compactions possible. The transformations are simple and easily manipulated. They are given in Figure 7.

Figure 8 shows the results of a compaction in a series circuit. The next step, not shown, would be to replace PATH1 and R1 with an equivalent parallel path. The compacted circuit could then be solved in qualitative terms or numerically.



7. Series (top) and parallel (bottom) transformations

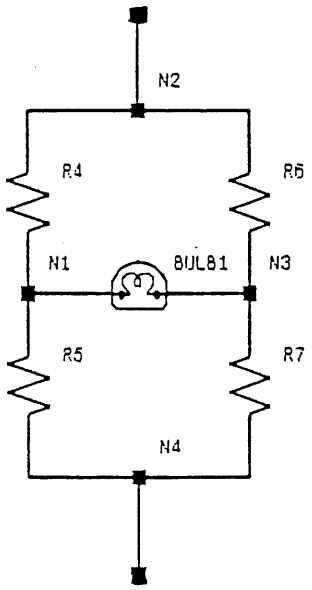


8. Component compaction into a series path

Doing the analysis in this manner does require redrawing the circuit, and perhaps computing several partial solutions. The student or the simulation never needs to remember more than a few variables. Once a student learns these transforms, he/she can analyze complicated circuits by reducing them to simpler circuits that he/she knows how to handle. Engineers will often compact a circuit of several elements. OREO does it two elements at a time for simplicity of coding and explanation. The two methods are equivalent, but because OREO groups elements into equivalent circuits two at a time, it may not generate exactly the same sub-circuits on its lowest level as the engineer does (who may group 3 or more elements at a time).

This method can be contrasted with the node or mesh analysis method which is more difficult, but also more powerful than beginners need. Nodal analysis requires the use of linear algebra and matrix manipulation. Equivalent circuit compaction keeps the math simple so it can be easily taught.

OREO compacts a circuit in this manner in order to orient it. As it compacts the circuit qualitatively, it could also lump the resistances and impedances together in a quantitative manner in order to perform a quantitative circuit analysis. As OREO unwinds the paths it created while orienting the subpaths, it could then analyze the circuit in a numeric way.



9. Wheatstone bridge circuit

As the circuit is reduced, further subcircuits appear. If there is a point where the series and parallel rules do not apply, but there is still more than one simple path across the voltage source, there is at least one bridge element[3] in the circuit. An example bridge circuit, the simplest one (known as a Wheatstone Bridge), is shown in Figure 9.

[3]A bridge element has a feed path and a ground path on both ends. Quantitative analysis is required to determine if there is a voltage drop, and its direction. The voltage drop across it is determined by the relationship between the two voltages created by the two voltage dividers. In the figure above, R4 and R5 make up one voltage divider, and R6 and R7 the other. In nearly all circuits bridge elements have a voltage drop across them.

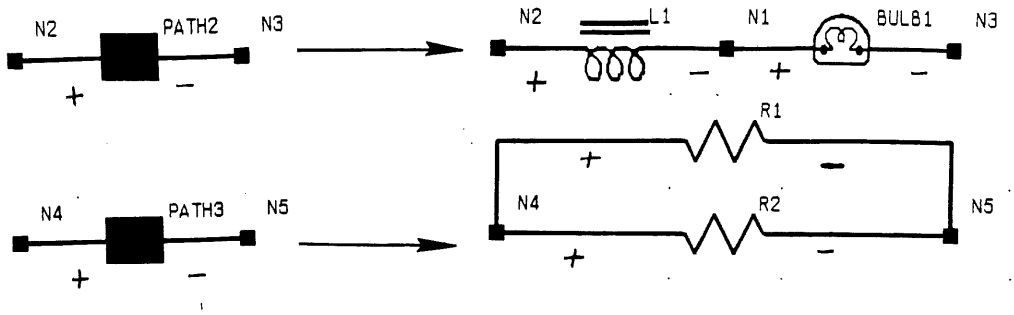
Initially we considered writing one rule for every bridge that the simulation should recognize. As students progressed they would learn to recognize more bridges, and thus would need more rules. We identified at least ten different bridges[4] that could possibly appear in troubleshooting. Teaching and programming bridge recognition in this way is not sound. It is easier to program and far easier to learn the general concept of a bridge, i.e., a path between two points, where each point is the middle of a voltage divider. This allows the real meaning of a bridge to be taught rather than rote pattern matching. It also underscores why one may need to deal quantitatively with bridges. The voltage drop across it is dependent on the relationship of two voltages that can not be qualitatively ordered. The best that one can do is to point out the bridge as a place for quantitative analysis. Most bridges are unbalanced, and few basic electrical components are directional. In the absence of either bridges or directional elements being explicitly in the circuit, engineers will assume an arbitrary direction for the bridge's orientation. OREO also arbitrarily chooses a direction for the voltage drop. (This assumes an unbalanced bridge, which is a reasonable assumption for most circuits.) A more formal definition of how OREO handles bridges is given in the Appendix.

For circuits that do not contain bridge elements, such as strict series-parallel circuits, decompression after compaction is easy: there will only be one path left at the top level. The sub-paths of the top-level path are decompressed using the orientation of the top level path. The inverse transformations are shown in Figure 10.

We have now shown the whole sequence of orienting a circuit. There are basically four steps:

- Note electrical nodes by compacting graphical nodes.
- Compact the series and parallel paths into their equivalent circuits.
- Orient the paths that are left.
- Orient by decompressing the paths using information from steps 2 and 3.

[4]Labelled aptly enough FrederiksenI-V and RitterI-V

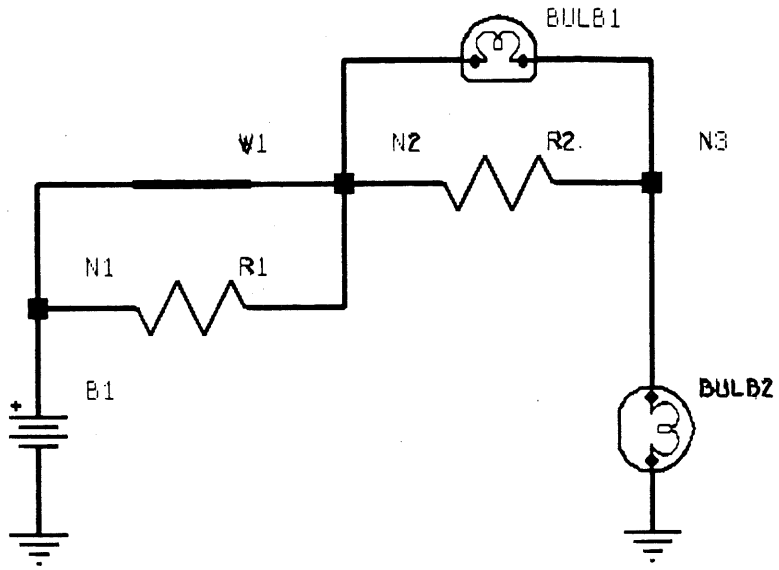


a) Series path expansion (top) b) Parallel path expansion (bottom)

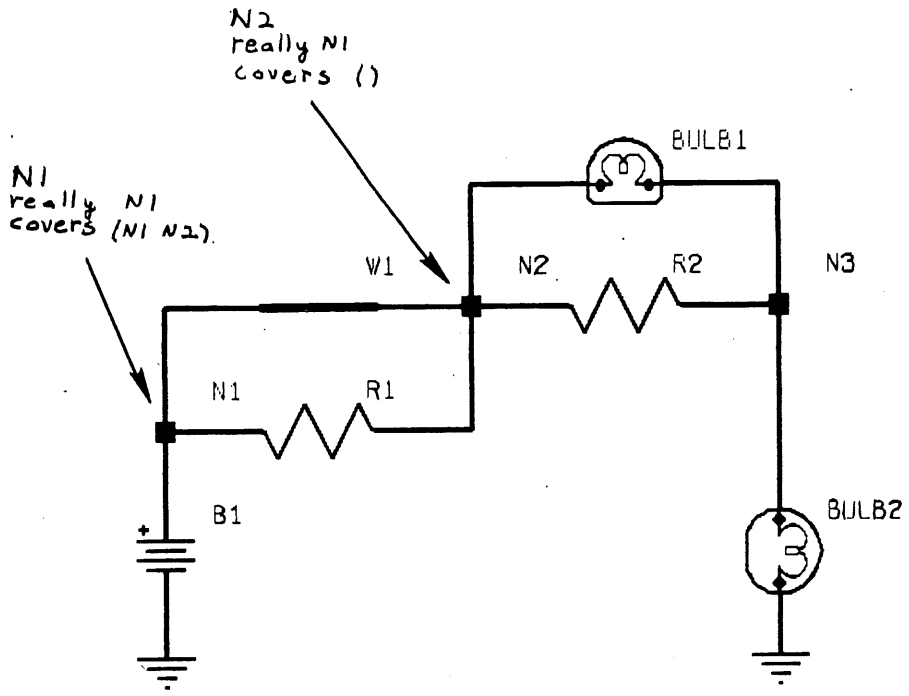
10. Inverse series and parallel transformations

3 Oreo At Work

The way OREO orients an example circuit, the one shown in Figure 11, is shown in the following figures. A summary of OREO's actions follows each step.

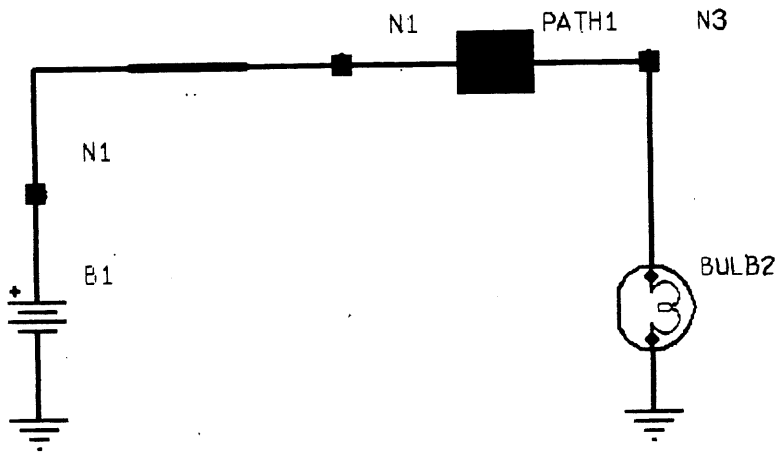


11. Original unoriented circuit



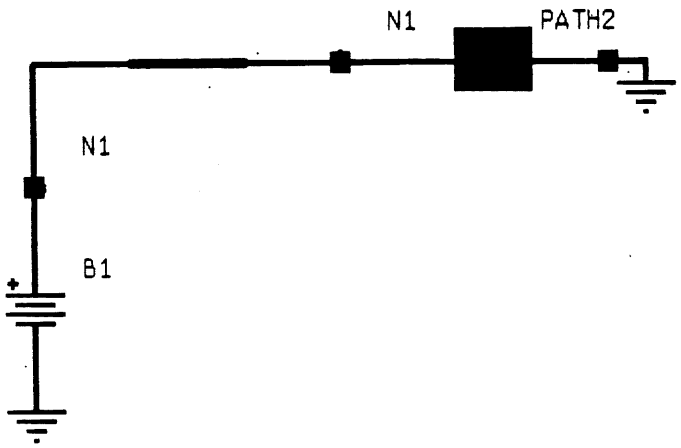
12. After Step 1 - Note electrical nodes

1. After finding B1 as the voltage source, OREO starts to group the graphical nodes into electrical nodes. N1 and N2 are connected together by a part that is conductive-non-resistive, W1. They are grouped together as N1. N3 and ground are not connected to other nodes through conductive-non-resistive elements, so they are left as themselves. R1 is now easily recognized as shorted because it is connected to the same electrical node on both ends. It will be ignored by later steps of OREO, as an engineer would. The results of this step are shown in Figure 12.



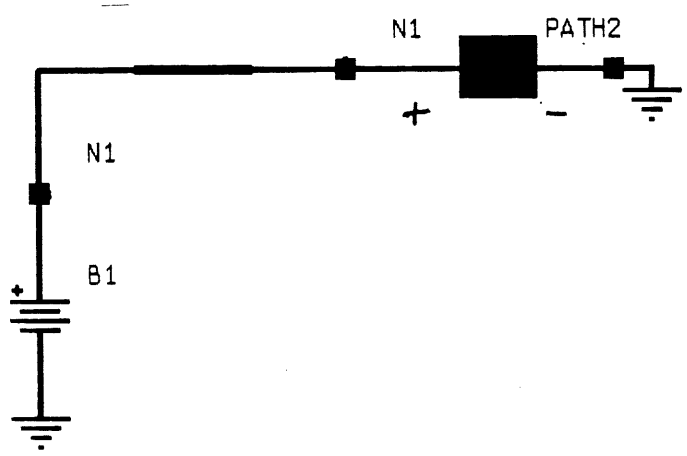
13. After Step 2 - Compact series and parallel circuits

2. Electrical node N1 is seen as a source of parts to compress into paths. B1 is excluded as a part for this analysis because it is the voltage source. There are two parts that can be considered connected to N1, BULB1 and R2. Because the parts go to the same node, it is a parallel path. BULB1 and R2 are replaced by PATH1, shown in Figure 13.



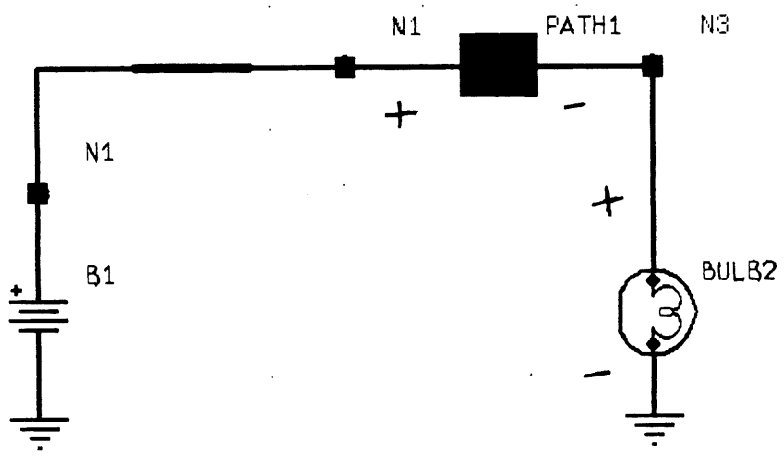
14. After Step 3 - Compact series and parallel sub-circuits (cont.)

3. N3 is checked next. There are two paths (a part is also considered to be a path) that terminate at different nodes, so Path1 and Bulb2 are made into a series path, PATH2. There are no more nodes left that have possible paths from them to compact, so path compaction terminates. The final result of compaction is shown in Figure 14.



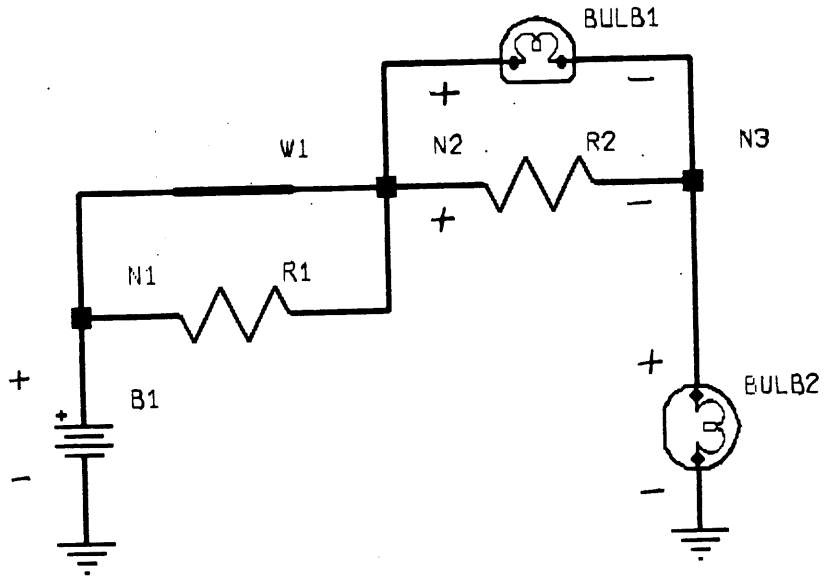
15. After Step 4 - Orient the paths

4. There is only one path to decompress. If there had been more, OREO would have oriented each one as explained in Section 2 and in the Appendix. In this case, the orientation of the path is straightforward. The end hooked up to N1 (which we know from Step 1 is hooked up to the voltage source) is set as a feed, and GND (the other node hooked up to B1) is the ground node. This is shown in Figure 15.



16. After Step 5 - Decompression

5. To decompress PATH2, BULB2 is oriented with respect to feed and ground, and PATH1 will be decompressed with N1 as feed.



17. After Step 7 - Orient the voltage source

6. BULB1 and R2 are oriented as the components of PATH1.

7. The voltage source B1 is reinserted, and the analysis is complete. The final circuit is displayed in Figure 17.

4 Implications for Qualitative Reasoning

Orienting circuits in this manner has several advantages. First, OREO allows QUEST to operate in a dynamic environment. Most current qualitative reasoning programs operate in static environments with the orientation of propagation fixed by the programmer [Holland, Hutchins, Weitzmann 84], [Rajagopalan 84], or done in a clever but unintelligible way to a student [Williams 84], [Stallman, Sussman 77]. The use of qualitative reasoning for engineering applications, such as design and simulation systems, will require the simulation to be completely automatic in a dynamic environment, and to be able to explain itself as QUEST does.

Second, OREO supports immediate propagation of constraints across devices that should be invisible to constraints. An example of this in the circuit domain is a switch closing; any change of voltage should propagate across the switch with practically no delay. While the switch is closed OREO removes it from all possible propagation paths by connecting the parts on each side of the switch to each other. Other systems don't appear to do this.

Third, because we are not only simulating circuit behavior, but also trying to teach students to reason about circuit behavior, OREO generates explanations designed to be easily understood by students. The path generation that occurs can be performed and understood by the student without using algebra or higher math. Analyzing the subcircuits in this way may be more tedious, but certainly can be more easily taught than forming and solving multivariable node or loop equations.

Fourth, based on its analysis, OREO has the potential to point out bridge elements to the student. Thus, OREO can handle loops in its constraint propagation space through its current default analysis, and students can learn about them also.

Finally, OREO and students can use these paths for quantitative reasoning. If there is a bridge, the elements that make it up have been reduced to equivalent circuit elements and it can be solved numerically. This is all the information available from qualitative methods. If further analysis is necessary, the data structures that are required for numeric analysis have already been built.

5 Future Directions

OREO was designed to orient the circuit from scratch, and it works well at that task. But experts do not discard the results of the previous analysis when reorienting a circuit. When circuit components change, OREO should be able to incrementally orient in a realistic manner. A first-pass improvement, yet to be implemented, is to reorient all parts in paths (recursively) connected to, or containing, the nodes affected when a part is added or changes connectivity (for example a switch closing), not just state or resistivity (for example a capacitor changing from uncharged to charging). The information necessary for this is saved on each part as the path that it is "part of." The algorithm seems complicated because of the computer representation of the circuit, but when viewed on the level of parts it actually is not. All it would need to do is look at the change of connectivity and the hierarchy of paths created on the last orientation.

This would make OREO more cognitively realistic, and give the entire system a better response time for the user. This inefficiency has not been a problem on small circuits, but in larger ones it can be. Orienting the circuit in Figure 11 takes OREO in compiled Interlisp less than 2 seconds running on a BBN Jericho work station[5].

6 Summary

In this paper we have shown the desirability of orienting electrical parts when analyzing a circuit qualitatively, for both students and computers. By proper initial orientation students can avoid many of the problems that hinder beginners. Qualitative computer simulations can now deal with dynamic circuit systems. OREO allows QUEST to avoid explaining backtracking which presents a more coherent and natural explanation of the analysis to the student.

Orientation also provides a useful framework for further levels of analysis by both students and computers. The paths that OREO generates and uses also provide useful mechanisms that would support quantitative simulation on either a subcircuit, or the whole circuit. If one wants to quantitatively analyze the whole circuit, this can be done in a manner which will be more easily understood by beginning students than the methods used in SPICE [Nagel 75].

[5]Comparable to a Xerox 1108.

ACKNOWLEDGEMENTS

The work that was presented here was developed through many and varied discussions with John Frederiksen, Barbara White, and Wally Feurzeig. OREO the program should be viewed as a porridge that had many cooks. Raman Rajagopalan, Eric Cooper, Jim Panagos, and Jim Schmolze provided useful comments. This work was supported by the Office of Navy Research and the Army Research Institute, under ONR contract N00014-82-C-0580 with the Personnel and Training Research Program.

Appendix A
IMPLEMENTATION NOTES

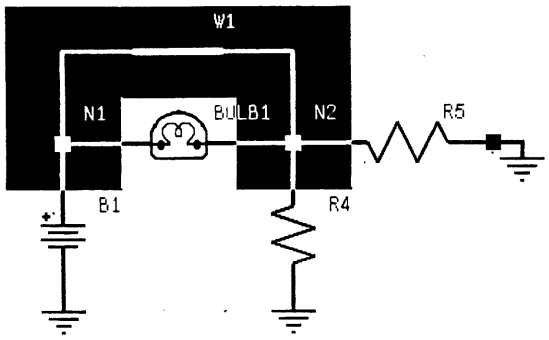
A detailed explanation of OREO's implementation is provided below. We will first detail the data structures, and then explain the algorithm used by OREO.

A.0.1 Data Structures in QUEST

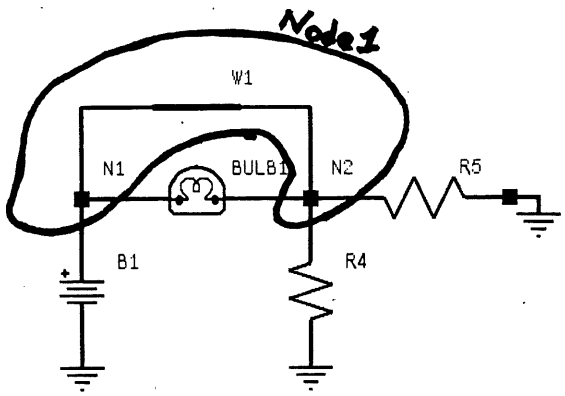
SPICE, perhaps the most significant quantitative electrical analysis program to date, is still in use today, and is still being improved [Deutsch, Newton 84]. It is typical of a whole family of quantitative electrical simulators. It stores its data only in node equations. QUEST represents nodes, parts, and paths as distinct types of objects. Nodes have slots for the part/port combinations that are connected to them. Because the distinction between electrical and graphical nodes is an important one, each node has a "really" slot which holds which electrical node it is a part of. If it is its own electrical node, any nodes which are "covered" by it are also saved. Saving this information would allow you to indicate visually the grouping of graphical nodes into the more useful concept of electrical nodes. Figure 18a shows how this pedagogical tool could look, and Figure 18b how it is typically drawn in electrical engineering texts.

All parts are represented as two-port devices and internal nodes. Each part internally saves its conductivity, connectivity, state, and voltage. Its general behavior rules are stored under its component type such as battery (B), or resistor (R). These rules can be found elsewhere [White, Frederiksen 86a]. Transistors and other multi-port devices could be represented as an expansion to connected two port devices. A list of the current components is presented in Table 1.

Paths represent equivalent subcircuits. They are marked with which type they represent, series, or parallel. Their end connections are also saved, and the parts or paths that make them up.



a) Possible presentation



b) Textbook type presentation

18. Grouping nodes into electrical nodes

Table A-1: Components Available in QUEST

	Abbreviation
Battery	B
Bulb	BULB
Capacitor	C
GND	(icon)
Inductor	L
Node	N
Points	POINTS
Resistor	R
Switch	SW
Test Light	LIGHT
Wire	W

A.0.2 Pseudocode

1. STRIP ALL MARKINGS FROM ALL PARTS AND NODES IN THE CIRCUIT
2. FIND THE GREATEST VOLTAGE SOURCE
3. NOTE ALL ELECTRICAL NODES

{In further analysis, only electrical nodes will be used to determine connections, or to trace paths. Being careful in this way ensures the minimum manipulation and reassignment of nodes. Because plus and minus on the battery (highest voltage source) are dealt with first, they will be separate and distinct e-nodes, even if there are several graphic nodes attached. This ability will reap dividends in pedagogy because the electrical nodes (plus and minus) of the battery will always be next to the battery.}

```
LOOP
 FOR nodes attached to the greatest voltage source,
     and then each graphical node in the circuit:
  IF the current node doesn't have an electrical node
     attached to it (in slot e-node)
  THEN
   . mark the current node as being its own electrical
        node
   . LOOP FOR every device attached to the node,
     IF the device is conductive-non-resistive
     THEN
     . join the two graphic nodes into one electrical
           node.
     . mark the part as being unoriented
       {It is shorted. This is currently implemented
        as no feeds and no grounds.}

To join two graphical nodes into one electrical
  node:

* IF the other node is an e-node, or has an
    associated e-node,
  THEN
  . set your e-node to be that e-node, it is older
  . set yourself to be covered by that e-node
  . IF you had become an e-node before you
       found this one,
    THEN
    .. set your covered nodes to be covered by that
          other e-node
    .. add your nodes to the its list of covered nodes
    ELSE
    .. set the e-node of the second node to be the
          e-node of the current node
    ENDIF

  ELSE
  . IF you are not part of an electrical node, or are
       not yourself an electrical node
    THEN  make yourself into an electrical node
    ENDIF
```

```
. set the other node to be a part of your
      electrical node
. mark the electrical node as covering the
       graphical node
```

4. COMPACT ALL SERIES AND PARALLEL SUBCIRCUITS

{A series-parallel circuit will compact into one path across the voltage source. If there are any nodes left to be put onto paths, it was not such a circuit. There are several options. The orientation algorithm can indicate that it has reached such a circuit, and apply extended bridge templates, or orient the parts and paths left with the next algorithmic step. OREO orients.}

```
For each e-node now in the circuit
      {i.e. each node that is its own e-node}
apply the following transformations:

   1.  If the node is of degree two,
          {We have already excluded parts that come
           back to the current node
       and the two paths go to different
       nodes, then make the two paths and the current
       node into a serial path joining the two nodes
       at either end.}

   2.  If the node is of degree 2 or more, find the
       nodes at the end of each path.  If any of
       them are the same, make those paths into a
       parallel path to that node.

If any of the transformations are used, reapply the
transformations, something may have become possible
that wasn't before.

Making a path:
  put the parts, their ports, and the intermediate
      nodes in order into a list,
  mark the node as being on a path now.
  mark what kind of path it is, which you will use
       when expanding the path.
  Where the parts had been hooked up, unhook them,
        and hook up the new path instead.
```

5. ORIENT THE PARTS AND PATHS LEFT (PART BY PART METHOD)

{The orientation of a path is just an inverse transformation. Figure 7 showed the transformations. If there is a feed path, and a ground path from a node, the node is a voltage divider. If a path or a part is connected to two nodes that have this property, it is the center element of a bridge, and it can not be oriented through purely qualitative means.

If you don't want to catch all bridges, but want to recognize the simple bridge configuration called the Wheatstone Bridge, you could apply the following transformation in the same way the series and parallel ones were applied in step 3.

First, the general strategy:

If a part has distinct[6] feeds and grounds (with respect to each other) for each possible orientation, then

[6]Distinct means that two paths do not share any nodes or sub paths.

it cannot be labelled, but surely has a voltage drop across itself in our simple world. We will arbitrarily choose a labelling. Otherwise, if it has at least one distinct feed (with respect to the chosen ground path) and at least one distinct ground (with respect to the chosen feed path) for only one orientation, then it is labeled with that orientation. If there is no such pair, it can not be labeled.}

The Algorithm:

```
for each part:
  for each pair of ports:
  let node1 = node port1 is connected to
      node2 = node port2 is connected to

  let feedpaths-n1 = feedpaths(node1)
      feedpaths-n2 = feedpaths(node2)
      gndpaths-n1  = groundpaths(node1)
      gndpaths-n2  = groundpaths(node2)

  for fpath in feedpaths-n1:
    for gpath in gndpaths-n2:
    if distinct(fpath gpath)
    then set one-way to be TRUE

  for fpath in feedpaths-n2
     for gpath in gndpaths-n1:
     if distinct(fpath gndpath)
     then set two-way to be TRUE

  if one-way and two-way
  then part is  a bridge element

  if not(one-way) and not(two-way)
  then part is unoriented

  if xor(one-way two-way)
  then orient the part based on which way.

{If you prefer to teach a rule to recognize a
 Wheatstone Bridge, the simplest bridge's rule is: }

IF * the node is of degree 3 or more, and
   * there are no further paths to compress, and
   * two paths go to distinct nodes, and
   * these two distinct nodes are of degree 3, and
   * are connected to each other by a single
       path, and
   * they are both connected to yet a third node,
THEN this set of nodes is a simple bridge path.
```

6. DECOMPRESS THE PATHS LEFT

{Decompress each path left using the orientation found in step 4. This uses the inverse transformations of Figure 10.}

7. ORIENT THE VOLTAGE SOURCE SO THAT IT FEEDS THE HIGHEST NODE

{When every other part has been done, finish up by orienting the voltage source in the direction

current (by convention positive) would flow. QUEST will trace the circuit in the direction the current flows; it will need to be able to pass through the voltage source and thus notice it.}

References

[Brown, Burton, Bell 74]
Brown, John Seely, Burton, Richard R. and Bell, Alan G.
Sophie: a Sophisticated Instructional Environment for Teaching Electronic Troubleshooting (an example of AI in CAI) Final Report.
Technical Report 2790, Bolt Beranek and Newman, 1974.

[Deutsch, Newton 84]
Deutsch, J. T., and Newton, A. R.
A Multiprocessor Implementation of Relaxation-based Electrical Circuit Simulation.
In *Proceedings of the 23 Design Automation Conference*, pages 350-357. IEEE, 1984.

[Holland, Hutchins, Weitzmann 84]
Holland, J.D., Hutchins, E.L. and Weitzmann, L.
STEAMER: an interactive inspectable simulation-based training system.
AI Magazine 5(2):, Summer, 1984.

[Nagel 75] Nagel, L. W.
SPICE2: A Computer Program to Simulate Semiconductor Circuits.
Technical Report ERL Memo No. ERL-M520, University of California, Berkeley, May 1975.

[Rajagopalan 84] Rajagopalan, Raman Meenakshisundaram.
Qualitative Modeling in the Turbojet Engine Domain.
Technical Report T-139, Coordinated Science Lab - University of Illinois, 1984.

[Stallman, Sussman 77]
Stallman, Richard M., and Sussman, Gerald J.
Forward Reasoning and Dependency-Directed Backtracking in a System for Computer-Aided Circuit Analysis.
Artificial Intelligence 9(2):135-196, October, 1977.

[Trick 77] Trick, Timothy.
Introduction to Circuit Analysis.
John Wiley and Sons, Inc., New York, 1977.

[White, Frederiksen 86a]
White, Barbara Y., and Frederiksen, John.
Intelligent Tutoring Systems Based upon Qualitative Model Evolutions.
In *AAAI-86: The National Conference on Artificial Intelligence*, pages 310-319. AAAI, 1986.

[White, Frederiksen 86b]
White, B. Y., and Frederiksen, J. R.
Progressions of Qualitative Models as a Foundation for Intelligent Learning Environments.
Technical Report 6277, BBN Laboratories, 1986.

[White, Frederiksen 87]
White, B.Y., and Frederiksen, J. R.,
Qualitative Models and Intelligent Learning Environments.
AI and Education ():, , 1987.
In press.

[Williams 84] Williams, Brian C.
Qualitative Analysis of MOS Circuits.
Technical Report 767, Massachusetts Institute Technology AI Lab, 1984.

@appendix(Appendix B: QUEST Troubleshooting Protocol)

In the following pages we briefly describe a short scene from a student troubleshooting a small circuit containing a battery, two resistors, a wire, and a bulb. This scene represents just a few minutes of such a session, which is itself only one of QUEST's (Qualitative Understanding for Electrical Systems Troubleshooting) modes of teaching.

The querying of the student that is done proceeds effortlessly for the student. He is asked before each action such as flipping a switch, or inserting a test light, what he hopes to learn through this action. The action is carried out by the system, and he may repeat this several times. After the effects of the changes are seen by the student calling for the simulation to be run, he is asked what he has learned. Each question need only, and can only be answered with the mouse. The student chooses from an appropriate range of responses. Some will take 2-3 clicks to indicate an intricate response, others are merely single clicks on a menu to indicate that he is exploring circuit behavior in a general way. The scene shown provides examples across this spectrum, including the extremes.

This interface is easy and natural to use. After the Session is over, there is clearly a substantial knowledge base developed of the student's plans and goals with a minimum of interference.

CC[illegible]Menu

Insert a test light

Flip a switch

Run the simulation

Insert a feed

Insert a ground

Remove a fault

Insert a fault

Insert a part

Change a connection

Delete a part

Quit

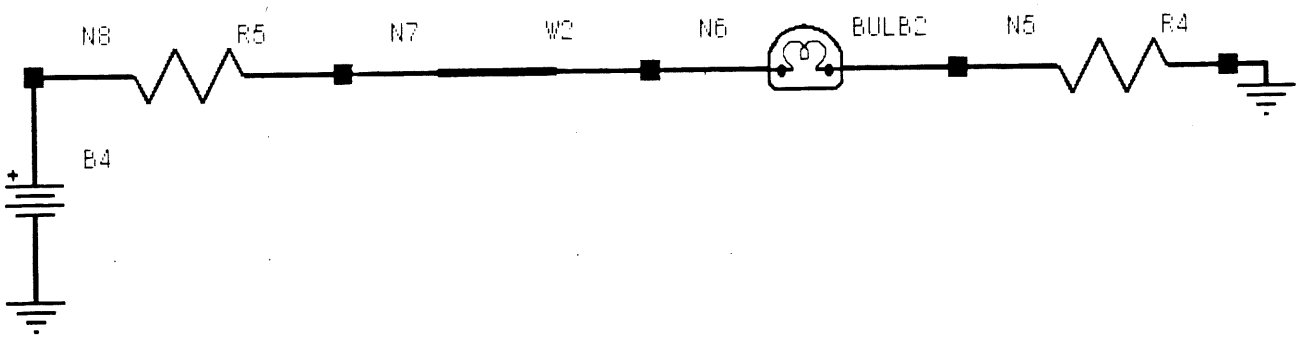
Go to the advanced menu

1

LOREN/QUEST Prompt Window

Trace Output

LOREN/QUEST Circuit Diagram



2 →

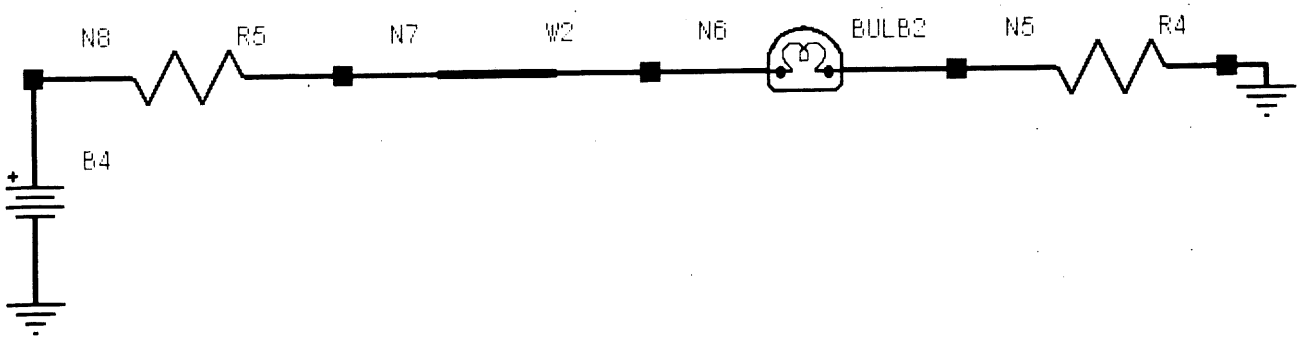
Don't Know
To explore general circuit behavior
To explore device behavior
To test device
To test feed to device
To test ground side of device

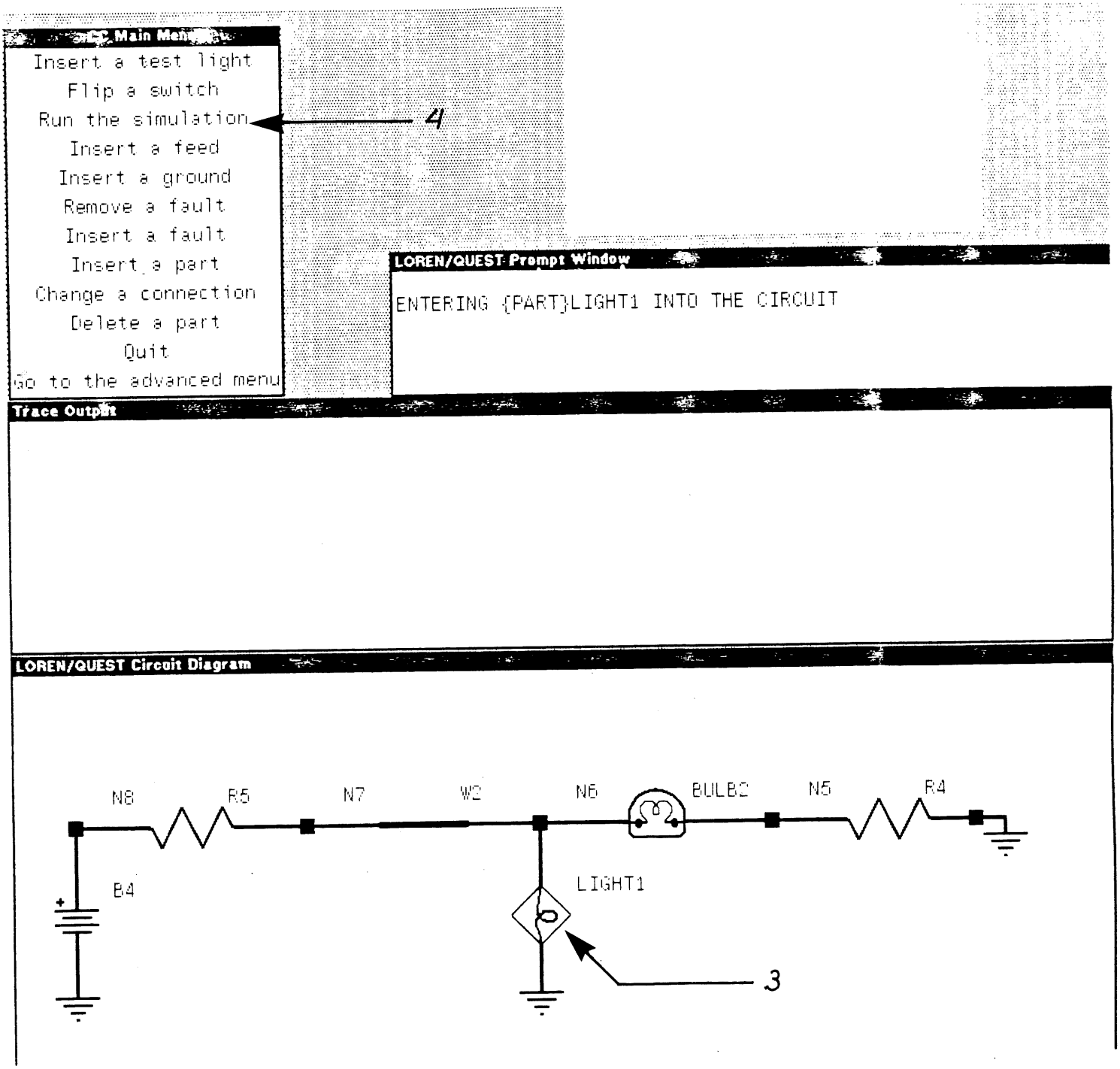
LOREN/QUEST Prompt Window

Please enter reason for action:

Trace Output

LOREN/QUEST Circuit Diagram




Main Menu
Insert a test light
Flip a switch
Run the simulation
Insert a feed
Insert a ground
Remove a fault
Insert a fault
Insert a part
Change a connection
Delete a part
Quit
Go to the advanced menu
4
LOREN/QUEST Prompt Window
ENTERING {PART}LIGHT1 INTO THE CIRCUIT
Trace Output
LOREN/QUEST Circuit Diagram
N8
R5
N7
W2
N6
BULB2
N5
R4
B4
LIGHT1
3

5

CONTINUE
STOP

6

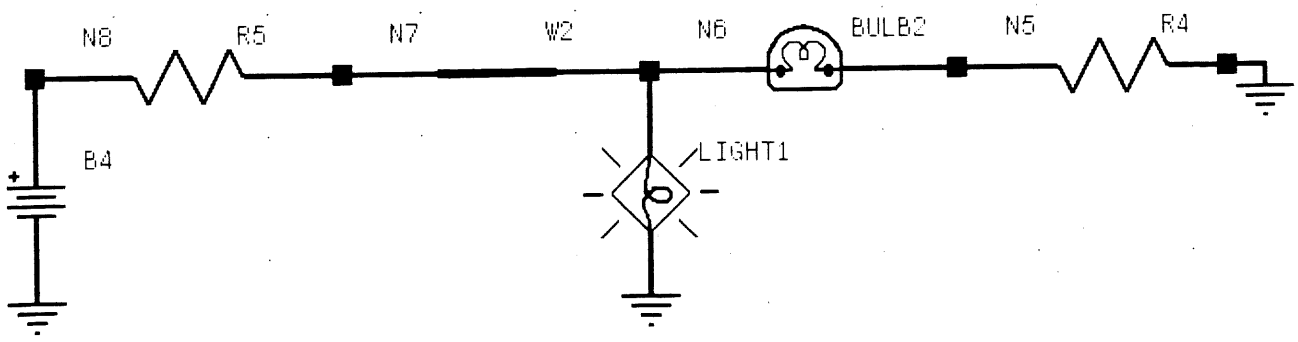
LOREN/QUEST Prompt Window

Continue simulation?

Trace Output

```
[...There is a voltage drop across part {PART}BULB2...]
[...Checking state rules for {PART}R4...]
[...RUNNING THE SIMULATION...]
[...TIME IS : 1...]
[...Checking time rules for {PART}LIGHT1...]
[...Checking time rules for {PART}B4...]
[...Checking time rules for {PART}R5...]
[...Checking time rules for {PART}W2...]
[...Checking time rules for {PART}BULB2...]
[...Checking time rules for {PART}R4...]
```

LOREN/QUEST Circuit Diagram



8

Don't know
Indicate new possible fault
No indication of new faults
Rule out possible fault
No new faults ruled out
Indicate suspect subcircuit
Indicate newly cleared subcircuit
Proceed to next action

LOREN/QUEST Prompt Window

What did you learn?

Possible faults

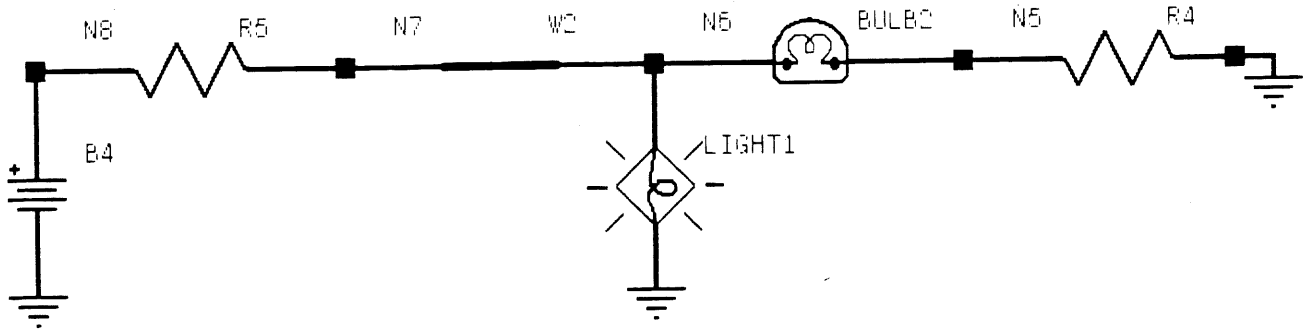
POSSIBLE FAULTS
NIL

Not possible faults

IN THE CLEAR
NIL

7

LOREN/QUEST Circuit Diagram



LOREN/QUEST Prompt Window

Part that is possibly faulted
CLICK FOR CHOICE

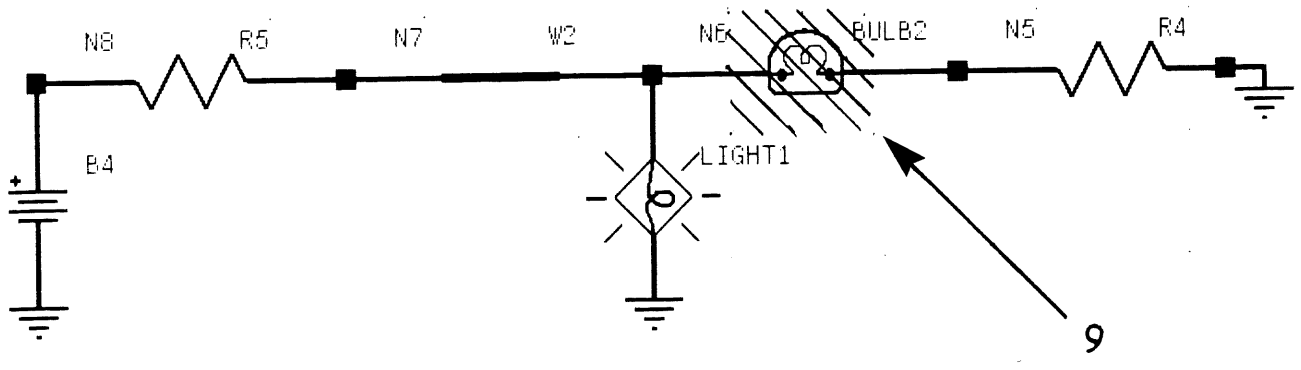
Possible faults

POSSIBLE FAULTS
NIL

Not possible faults

IN THE CLEAR
NIL

LOREN/QUEST Circuit Diagram



10

BULB Faults

- OPEN
- SHORT2GND
- SHORT

LOREN/QUEST Prompt Window

Indicate fault of BULB2

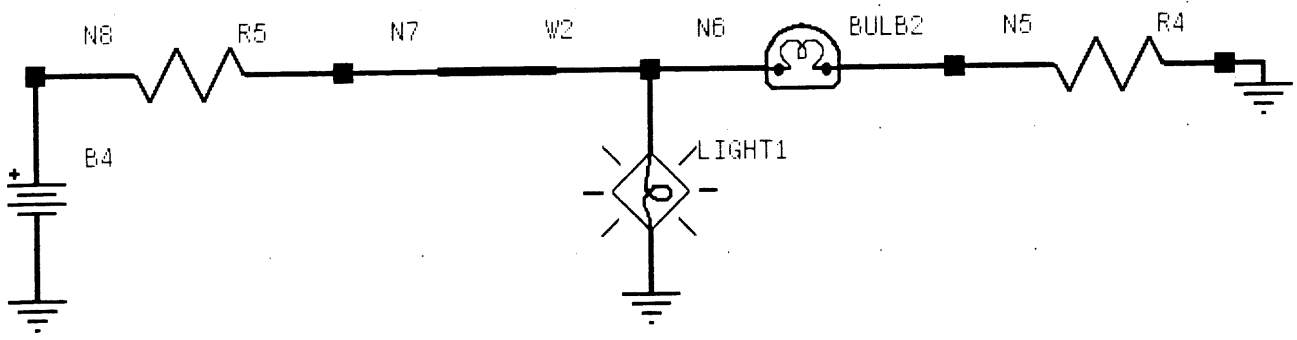
Possible faults

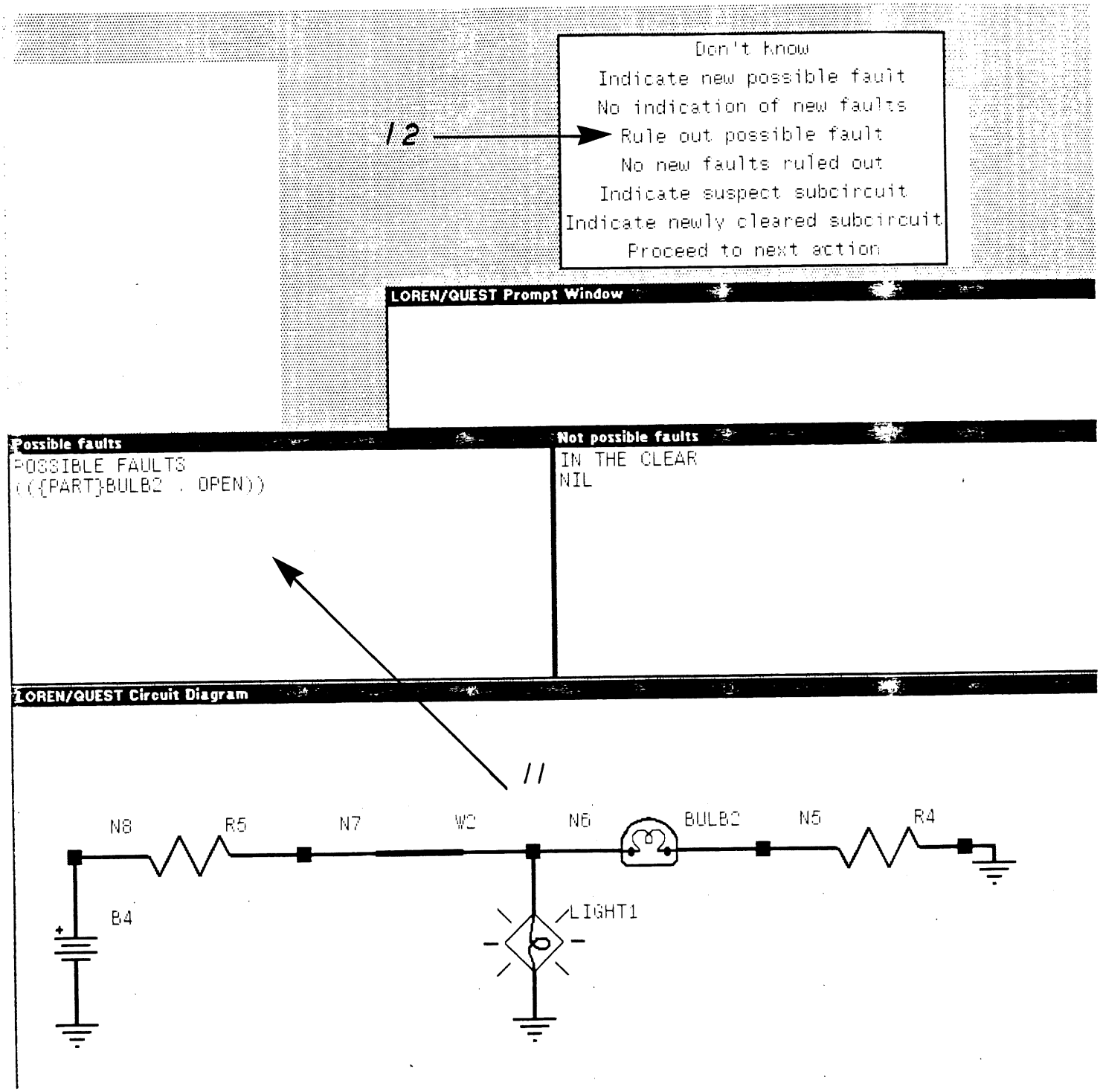
POSSIBLE FAULTS
NIL

Not possible faults

IN THE CLEAR
NIL

LOREN/QUEST Circuit Diagram




Don't know
Indicate new possible fault
No indication of new faults
12
Rule out possible fault
No new faults ruled out
Indicate suspect subcircuit
Indicate newly cleared subcircuit
Proceed to next action
LOREN/QUEST Prompt Window
Possible faults
POSSIBLE FAULTS
(({PART}BULB2 . OPEN))
Not possible faults
IN THE CLEAR
NIL
LOREN/QUEST Circuit Diagram
//
N8
R5
N7
W2
N6
BULB2
N5
R4
B4
LIGHT1

LOREN/QUEST Prompt Window

Part that is not faulted
CLICK FOR CHOICE

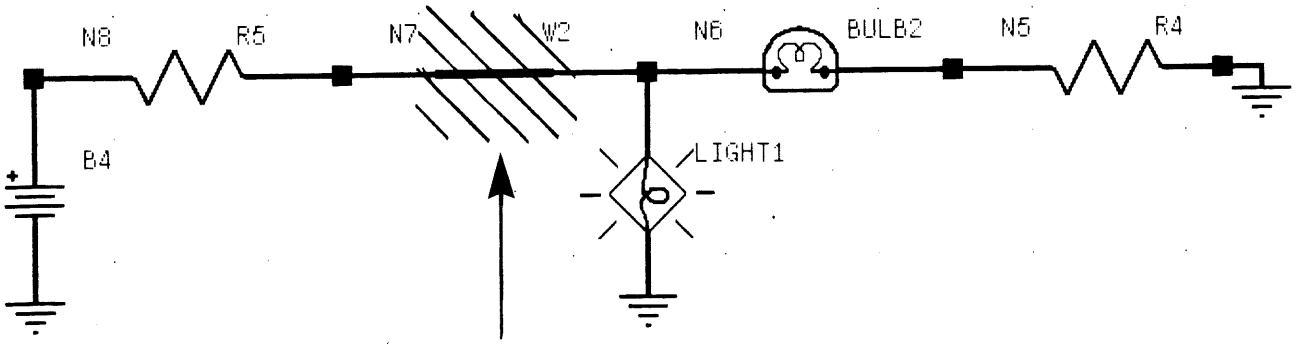
Possible faults

POSSIBLE FAULTS
(({PART}BULB2 . OPEN))

Not possible faults

IN THE CLEAR
NIL

LOREN/QUEST Circuit Diagram



Don't Know
Indicate new possible fault
No indication of new faults
Rule out possible fault
No new faults ruled out
Indicate suspect subcircuit
Indicate newly cleared subcircuit
Proceed to next action

14

LOREN/QUEST Prompt Window

Indicate fault of W2

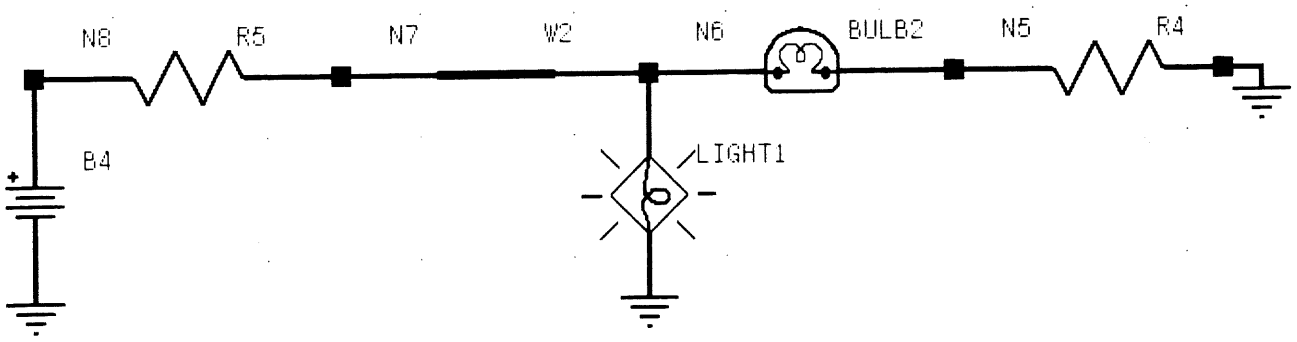
Possible faults

POSSIBLE FAULTS
(({PART}BULB2 . OPEN))

Not possible faults

IN THE CLEAR
(({PART}W2 . OPEN))

LOREN/QUEST Circuit Diagram



CC Main Menu

Insert a test light
Flip a switch
Run the simulation
Insert a feed
Insert a ground
Remove a fault
Insert a fault
Insert a part
Change a connection
Delete a part
Quit
Go to the advanced menu

LOREN/QUEST Prompt Window

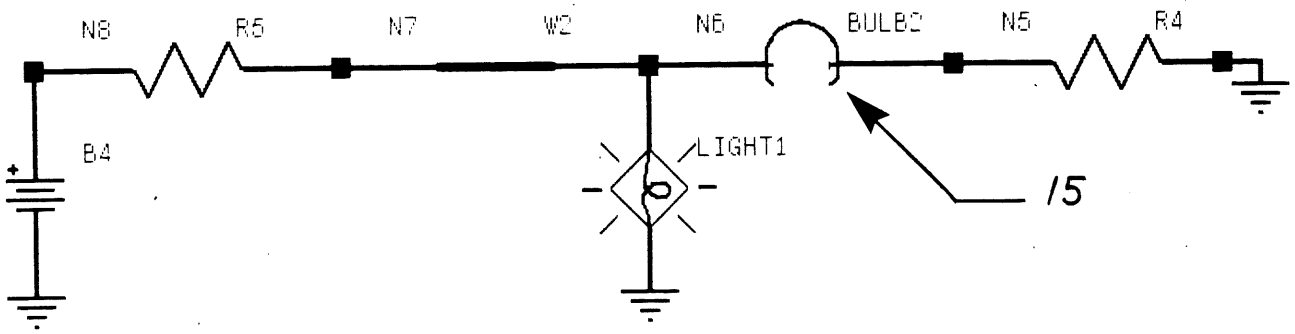
Possible faults

POSSIBLE FAULTS
(({PART}BULB2 . OPEN))

Not possible faults

IN THE CLEAR
(({PART}W2 . OPEN))

LOREN/QUEST Circuit Diagram



Prompt Window