A multi-level, multi-purpose qualitative network flow analysis technique

Richard C. Shipman

Department of Computer Science
University of Wales
Aberystwyth

September
2011

This thesis is submitted in partial fulfilment of the requirements for the degree of

Master of Philosophy of The University of Wales.

Declaration

This thesis has not previously been accepted in substance
for any degree and is not being concurrently submitted in candidature
for any degree.

Signed .... (candidate)

Date ....

Statement 1

This thesis is the result of my own investigations, except
where otherwise stated.

Other sources are acknowledged by footnotes giving explicit
references. A bibliography is appended.

Signed .... (candidate)

Date ....

Statement 2

I hereby give consent for my thesis, if accepted, to be made
available for photocopying and for inter-library loan, and for the
title and summary to be made available to outside organisations.

Signed .... (candidate)

Date ....

ABSTRACT

Research at Aberystwyth into qualitative simulation of electrical circuits has been word-leading for nearly two decades. During that time, the Advanced Reasoning Group has developed a variety of electrical circuit simulators. The most successful, CIRQ, has been deployed for more than ten years in industrially useful design analysis systems, and has undergone a range of improvements over that time.
MCIRQ is a more advanced version of CIRQ that was developed on the *Design for the Whole Vehicle, Whole Lifecycle* EPSRC Project at Aberystwyth. It is able to reason about circuits with multiple levels of qualitative resistance, and to indicate the different levels of current activity in the circuit. It showed great promise of being able to assist the design analysis systems in producing improved analysis reports, but was never deployed industrially because it was not able to analyse as wide a set of circuits as the original CIRQ software.
This thesis describes a reimplementation of MCIRQ for use with both electrical and fluid flow systems. It extends the theory of electrical qualitative analysis in several ways, and makes several improvements to the original version of MCIRQ:

- It replaces the Forward-Reverse labelling system of path finding with Series-Parallel reduction system to determine current magnitude and direction.
- It improves the Star-Delta Reduction algorithm to deal with problem cases that do not occur in well-formed electrical circuits, but which do occur in the circuits with failure that are dealt with in failure analysis work.
- It extends MCIRQ to deal with many bridge circuits that were impossible to resolve in the previous version of MCIRQ.
- It introduces a principled strategy for resolving the direction of current flow within the circuit.

The overall effect of these improvements is to transform MCIRQ into a qualitative analysis tool that can be used for real world applications. The thesis illustrates the kinds of analysis that can now be done by MCIRQ that was impossible before. The new MCIRQ has been employed on the ASTRAEA project. It has made possible applications of design analysis for engineering systems that could not have been addressed by any of the previous circuit analysers, notably in the domain of systems with both electrical and hydraulic aspects.

ACKNOWLEDGEMENTS

My thanks to all the staff in the Computer Science Department, but especially to Dr. Neal Snooke, my supervisor, and Prof. Chris Price, both of whom have supported me throughout this process.

To my wife Suzy, and children Benjamin and Abigail for putting up with me working on this for so long.

Contents

List of Figures

List of Tables

Code/Data fragments

List of abbreviations

ASTREA	Autonomous Systems Technology Related Airborne Evaluation & Assessment
AutoSteve	An automated FMEA package developed by FirstEarth Ltd using CIRQ
CAD	Computer Aided Design
CIRQ	A pathfinding circuit solver using 3 levels of resistance
F/R	Forward-Reverse pathfinding
FMEA	Failure Mode Effects Analysis
IDE	Integrated Development Environment
M^2CIRQ	Multiple Source Multiple level CIRQ
MBD	Model Based Diagnosis
MBR	Model Based Reasoning
MCIRQ	A development of CIRQ that uses multiple levels of resistance
O-O	Object-Oriented
QR	Qualitative Reasoning
S-P	Series-Parallel reduction
SPICE	Simulation Program with Integrated Circuit Emphasis
TUM	Technical University of Munich

Chapter 1

Introduction

This thesis presents a new method for determining flow in an electrical circuit which allows multiple qualitative values for resistance, resulting in qualitatively different values of flow in the circuit. This method is encapsulated in an algorithm for analysing the behaviour of electrical circuits more effectively, and in a more principled manner. This algorithm has been implemented and integrated within a model-based reasoning system which performs automated failure modes and effects analysis (FMEA) and diagnosis of automotive electrical systems.

The resultant system has been tested on electrical circuits which gave problems with previous algorithms, and has shown itself to be capable of overcoming those problems. Furthermore, the algorithm has also been used with hybrid hydraulic/electrical systems, and also proves effective in that domain.

The main contributions of this work are extensions to Series-Parallel Reduction to deal with cases that are not covered by the basic theory, and replacement of an ad hoc approach to derivation of current flow in a circuit with a principled approach to current flow analysis that gives greater confidence that current flow results will be correct for large, complex circuits.

1.1 Background

The advanced reasoning research group at Aberystwyth has a long history of successful research in model-based reasoning for electrical automotive systems [Price and Lee, 1988, Hunt et al., 1993, Struss and Price, 2003]. One of the foundations of that research is the ability to perform Qualitative Circuit Analysis. The original algorithm to achieve this was developed by Prof Mark Lee on the SERC Project "DREAM" [Lee and Ormsby, 1992, Lee and Ormsby, 1993], and was gradually refined to meet the demands of real world case studies [Lee, 1999, Lee, 2000]. It still has some limitations which need to be addressed, primarily with *bridge circuits* [Lee et al., 2001].

The work presented in this thesis addresses some of the shortcomings of the previous qualitative circuit analysers. It is based on Series-Parallel (S-P) reduction [Mauss and Neumann, 1996] rather than Forward-Reverse (F/R) pathfinding [Lee et al., 2001], and includes the ability to perform electrical analysis on a system with multiple levels of resistances rather than the three levels present in the original CIRQ system [Lee and Ormsby, 1993]. This can involve the automatic generation of new current values that do not directly map into the quantity space that would be directly derived from the set of resistances used in the circuit. The new current values are used when parallel current paths of different resistances combine in such a way as to divide current unequally across two paths.

The use of multiple values for resistance, linked to the use of Series-Parallel reduction shows promise in the simulation of a wider variety of systems and domains. The technique also shows promise for other forms of flow based simulation such as in the fluid flow domain. The development of a new method for qualitatively solving flows in circuits with multiple values of resistance is the main result of this dissertation. The implementation and testing of a multiple qualitative value flow simulator incorporating these ideas has also been achieved.

The simulator based on the work presented here now forms the core qualitative simulation engine used for Failure Effect Analysis as part of the ASTRAEA project both for electrical and fluid flow systems as presented at ISSRC in Singapore in 2008 [Snooke et al., 2008]. The ASTRAEA project is a £30M Technology Strategy Board initiative to develop the safety case for unmanned aerial vehicles flying in commercial airspace [Downes, 2007].

1.2 Content of this thesis

This thesis is structured in the following way:

Chapter 1 gives a brief summary of the content of the thesis, and explains where it fits into a larger body of research at Aberystwyth. It also includes this summary of the contents of each chapter.

Chapter 2 is a survey of relevant literature. It focuses on qualitative reasoning and its application to engineering problems. As the main focus of this thesis is on an improved method for simulating electrical circuits, the literature survey concentrates on previous useful methods for qualitative simulation of electrical circuits.

Chapter 3 gives details of the improvements that can be made to the previous CIRQ and MCIRQ systems in order to produce better simulation results in a more principled way.

Chapter 4 provides some details of the implementation of the improved MCIRQ system.

Chapter 5 examines the performance of the improved MCIRQ system on circuits which proved problematic for the previous versions of the CIRQ and MCIRQ simulators. It also describes the successful deployment of the new simulator within the ASTRAEA project.

Chapter 6 rounds off the thesis with conclusions and future work.

Chapter 2

Literature Review

2.1 Model-based Reasoning

The phrase *model-based reasoning* has been used to refer to the task of using models of a system to reason about the behaviour of the system being modelled [Milne, 1991]. The types of model that are used should reflect the structure of the domain in a fairly direct way, often modelling components and connections between the components in the domain. Much of the early work in model-based reasoning was for the task of diagnosis, and was specifically referred to as model-based diagnosis (MBD). The general idea was that the model of the system predicts what should be happening in the observed system, and that differences between the model of the system and the observations of the real system could be used to monitor the real system, and to perform diagnosis when there were discrepancies.

Qualitative reasoning [Bobrow, 1985] has often been used as a basis for model based reasoning, as it gives excellent coverage of the possible behaviours of a system [Travé-Massuyès and Milne 1995, Price et al., 2006].

[Travé-Massuyès and Milne 1995] identifies six key areas of application for qualitative model-based reasoning:

- continuous processes
- engineering

- ecology
- electronic circuits
- business and commerce
- medicine

While not all of these areas are of the same type (for example, electronic circuits could clearly be classified under engineering), they give some idea of the range of areas where this technology is being applied.

[Price et al., 2006] is based on a roadmap for the future of qualitative reasoning produced by the MONET Network of Excellence in Model-based and Qualitative Reasoning funded by the European Commission, and gives a summary of the application areas of model-based and qualitative reasoning. Looking back at the past achievements of qualitative and model-based reasoning, it chooses to highlight 10 significant areas of development:

- Fault detection from models

  This has been applied in automotive systems [Struss and Price, 2003], in monitoring of large gas turbines [Travé-Massuyès and Milne 1997] and in spacecraft diagnosis [Williams and Nayak, 1996].

- System simulation using virtual prototypes

  The area of virtual prototyping is an important application of model-based reasoning - it saves the significant cost of implementing actual systems in order to identify potential problems with a design by instead performing tests on the model-based system. [Price et al., 2006] discusses the work described in [Ward and Price, 2001], which is one aspect of the automotive work at Aberystwyth, along with [Bénazéra and Travé-Massuyès, 2003] which uses models to produce efficient diagnostics before an actual satellite system has been built.

- Process understanding and monitoring

  While the models here tend to be process-based rather than structural component based, the intuition behind the use of models is similar. Examples are given of chemical process plants where models of the system enable improved monitoring and control of that system.

- Explanation of numerical simulations

  [Forbus and Falkenheiner, 1991] show how qualitative and quantitative simulation can be linked to provide simulators that are capable of explaining how they reached their results. The functional reasoning layer of the AutoSteve system described later in this chapter [Price, 1998] performs similar abstraction of detail to a level understandable to the user.

- Cognitive applications

  [Price et al., 2006] argues that these kinds of model can also be used to better understand peoples' models of real world systems, but that is outside the considerations of this thesis.

- Compositional model-based diagnosis and state tracking

  [Price et al., 2006] talks about the use of model-based systems with online observations to track the overall state of a system, and to alert the user to incipient problems. The work described in this thesis has been used within ASTRAEA for this type of application (not yet published, but covered by BAE Systems patent applications 0910145.2 and 1107160.2).

- Model-based systems provide many opportunities for re-usability

  This is an advantage of model-based reasoning rather than an application area. The work in the automotive area very much relies on this advantage, with reusability at the level of functional description, of circuit component, and of primitives for the underlying qualitative circuit analysis [Struss and Price, 2003].

- FMEA generated from the design description and component models

  FMEA is one of the key application areas of the automotive model-based reasoning discussed in this thesis [Price and Taylor, 2002], and the work described in this thesis has made it possible to generate FMEA consequences for situations that would previously been unresolved because of shortcomings in the qualitative circuit analysis.

- QR models in the educational context

  [Price et al., 2006] highlights the use of models to teach people about how things work, but again that is outside the considerations of this thesis.

- QR to help decision making under uncertainty

  QR can be used where numerical approaches are not applicable. This is somewhat outside of the scope of this thesis, and so is not considered further.

2.2 Structure of a Model-based FMEA system

AutoSteve is a software package that performs a number of analyses on electrical circuits, initially developed in the Computer Science department at Aberystwyth University and subsequently developed, marketed and sold by FirstEarth Ltd. This section looks in detail at the way that the AutoSteve system is structured to perform FMEA. The qualitative circuit simulator that has been designed and implemented in this thesis forms part of the structure of the AutoSteve software.

FMEA is a design discipline normally performed by a group of engineers who, between them, understand all of the aspects of the design of the system being analysed. They consider the ways in which the system can fail, and they explore the consequences of every possible failure, and assess its significance. The purpose of this exercise is to highlight potential catastrophic failures of the system, and to amend the design to avoid them if possible.

Automated FMEA of electrical systems considers the effects of every possible component failure within an electrical system, exercising the behaviour of the system containing the fault, and reporting the differences between the failure behaviour and the correct behaviour. If the behavioural differences were reported at the level of individual resistances or individual components, then the significance of the report would be lost in a plethora of details. Therefore, it is necessary to abstract the results to an appropriate level for an engineer to understand.

The automated FMEA system is made up of three layers:

1. An electrical qualitative analyser. This is the layer where the system built as part of this thesis is included in the overall FMEA system. It takes a grid of resistors and connections, and decides on current flow within the grid.

2. A state based component reasoner. This layer works at the level of the components drawn by an engineer in a CAD tool. Given the state of each component in the system, this layer interacts with the electrical qualitative analyser to decide how the system will respond to any input.

3. A functional reasoning layer. This layer takes the results of component-based simulation and abstracts them to the level appropriate for presentation to an engineer.

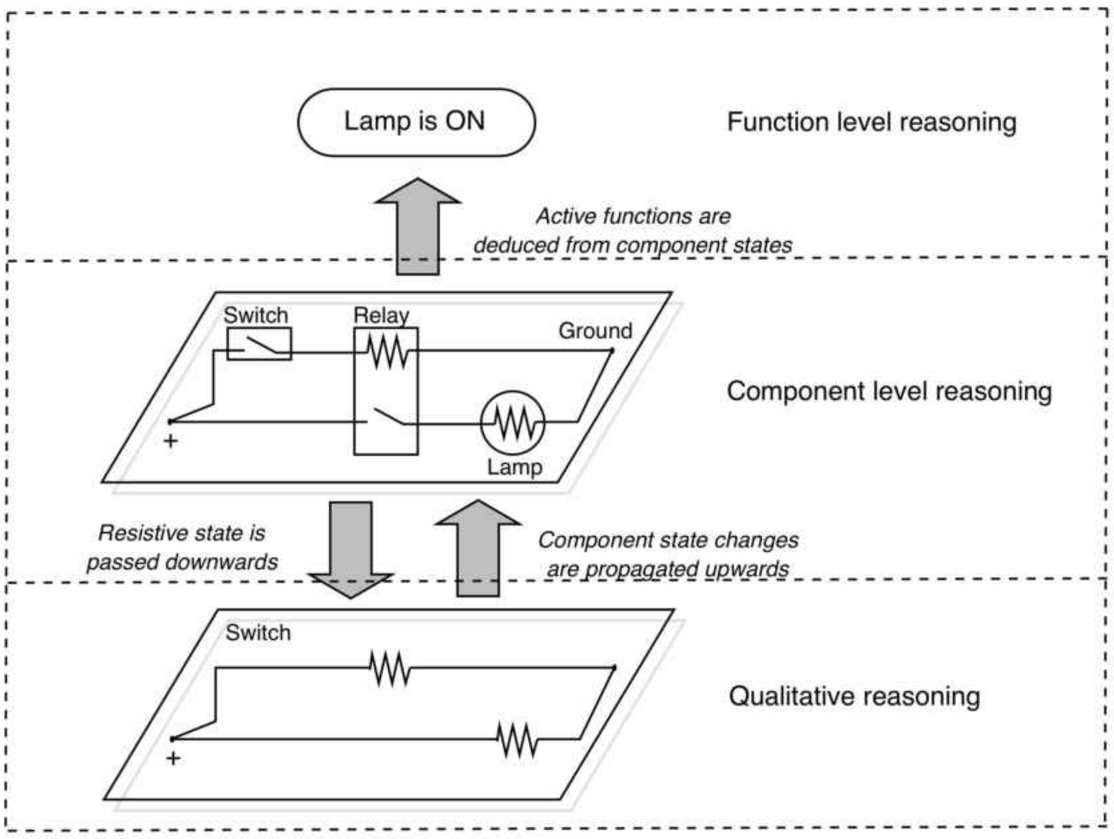

Lamp is ON
Function level reasoning
Active functions are deduced from component states
Switch
Relay
Ground
+
Lamp
Component level reasoning
Resistive state is passed downwards
Component state changes are propagated upwards
Switch
+
Qualitative reasoning

Figure 2.1: Layers of the Model-Based FMEA System

Figure 2.1 shows how these three layers relate for a very simple lighting circuit. The detail of how the AutoSteve system works is not germane to this thesis, it is merely presented as context for the electrical qualitative analyser. If the reader is interested in the details of the AutoSteve system, then a number of papers have been written on this subject, notably [Price, 1998, Price, 2000, Price and Taylor, 2002, Price et al., 2003].

The electrical qualitative analyser is the underlying engine for all of this work — it is exercised many thousands of times during an FMEA analysis, and needs to be both efficient and effective. When the first prototype was built at Aberystwyth, performing an FMEA took several days. This was improved by reimplementing the electrical qualitative analyser in a more efficient language, and identifying others ways that the code could be improved. The electrical qualitative analyser also needs to be able to predict current flows for as many circuit configurations as possible - where analysis comes up with ambiguous results for a single component state, then producing an FMEA is not possible for that component failure.

The original electrical qualitative analyser used in AutoSteve was CIRQ (see section 2.4.1 for details). [Price et al., 2003] explores how CIRQ can be replaced by other electrical circuit analysers. This is shown in Figure 2.2, taken from [Price et al., 2003] with permission.

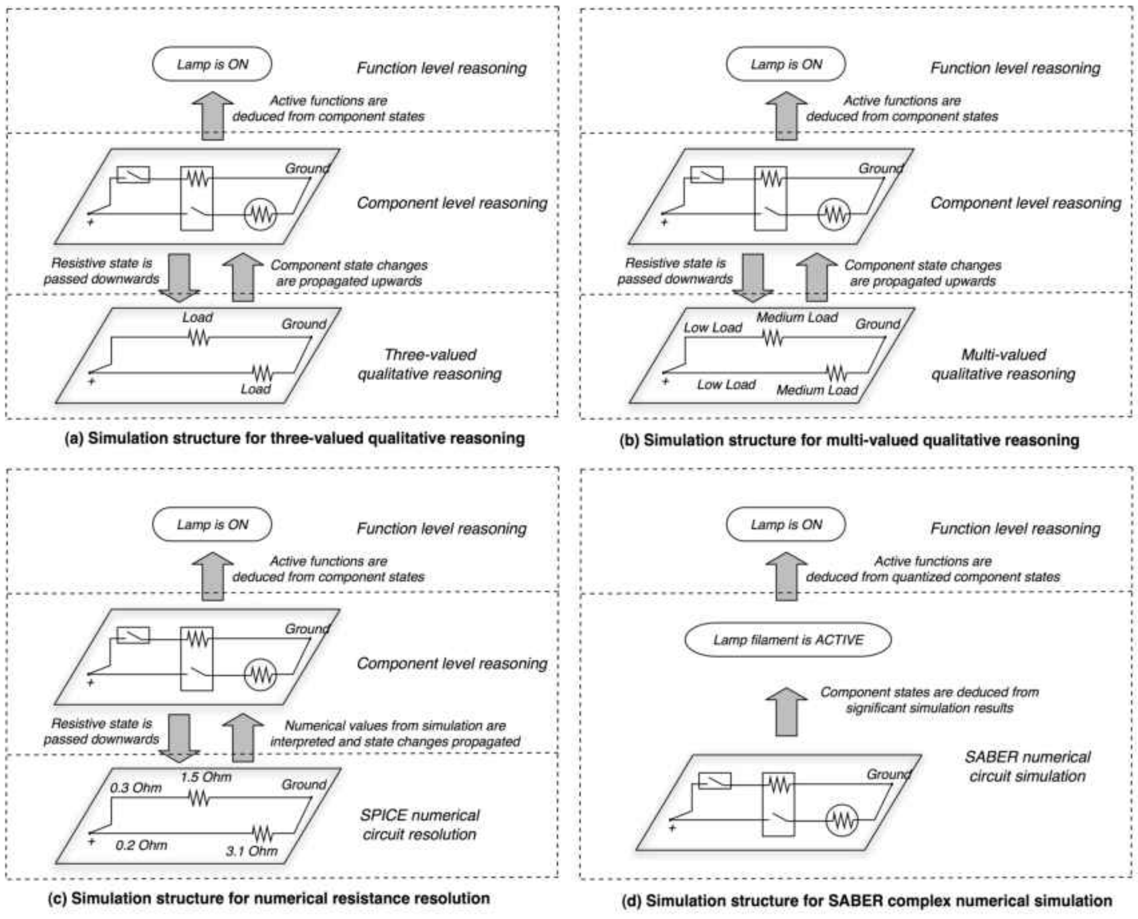

Lamp is ON
Function level reasoning
Active functions are deduced from component states
Ground
Component level reasoning
Resistive state is passed downwards
Component state changes are propagated upwards
Load
Three-valued qualitative reasoning
(a) Simulation structure for three-valued qualitative reasoning
Low Load
Medium Load
Multi-valued qualitative reasoning
(b) Simulation structure for multi-valued qualitative reasoning
Numerical values from simulation are interpreted and state changes propagated
0.3 Ohm
1.5 Ohm
0.2 Ohm
3.1 Ohm
SPICE numerical circuit resolution
(c) Simulation structure for numerical resistance resolution
Active functions are deduced from quantized component states
Lamp filament is ACTIVE
Component states are deduced from significant simulation results
SABER numerical circuit simulation
(d) Simulation structure for SABER complex numerical simulation

Figure 2.2: Use of different simulators in producing functional level reports: (a) simulation structure for three-valued qualitative reasoning; (b) simulation structure for multi-valued qualitative reasoning; (c) simulation structure for numerical resistance resolution; (d) simulation structure for SABER complex numerical simulation.

Part (a) of the figure shows the AutoSteve system already described, with CIRQ as the electrical qualitative analyser. Part (b) of the figure shows CIRQ replaced by MCIRQ (described in section 2.4.4. Some adaptation of the interface between the state-based component reasoner and the electrical qualitative analyser is needed in order to handle the multi-level current flows given as results from MCIRQ, but the system is essentially unchanged. Part (c) of the figure shows the electrical qualitative analyser being replaced by SPICE. Again, some adaptation is needed to handle the more precise results that are available from SPICE, but the system essentially works in the same way. Part (d) of the figure is somewhat different, as the the state-based component reasoner is replaced as well as the electrical qualitative analyser, but this configuration was always less successful than the ones that merely replaced the electrical qualitative analyser.

The new electrical qualitative analyser would fit in as a replacement for CIRQ in the same way as is done in parts (a,b,c) of Figure 2.2. The adaptations needed to do so are similar to those needed for MCIRQ, as this analyser also produces multi-level current flows.

2.3 Other Applications in Electrical Domains

One of the major advantages of building a system such as AutoSteve on top of electrical qualitative analysis has been that the results have been general enough that they have been applicable to a range of other analysis tasks in the automotive domain.

A number of other analysis tools have been built by the research group at Aberystwyth over the years. Many of them now form part of the design analysis suite sold by Mentor Graphics for their Harness Design tools [Mentor Graphics website, 2011], although the latest version of several of these tools have been developed by Aberystwyth with more advanced functional abstraction, more complex component representation, and using the qualitative circuit analyser described in this thesis.

2.3.1 Workshop diagnosis manual

This application does not involve any different analysis from that done for the FMEA. Instead, it takes the FMEA results, and rearranges the results ordered by symptom rather than component failure. It then can tell the mechanic which possible component faults could be responsible for the failure of specific functions such as losing the use of a main headlight. These potential causes can be shown to an engineer on a circuit diagram, enabling him to explore what is wrong and fix it.

2.3.2 On-board diagnosis

This is more complex than workshop diagnosis. The task is to use on-board measurements to monitor the system, and to detect of the occurrence of faults. Ideally, the system would also enable isolation of the fault, as least to some degree. The latest version of the FMEA has been used on the ASTRAEA project to produce on-board diagnostics for the fuel system for an unmanned aerial vehicle, as pictured in Figure 2.3. The fuel system contains hydraulic components as well as electrical components, and both aspects of the fuel system are modelled in the three layer system described earlier, and use the qualitative analyser described in this thesis at the lowest level.

The diagnostics produced are integrated within a Bayesian diagnostic system built by BAE Systems, and can be used both to implicate and to exonerate component faults. No paper is presently available detailing this work, as it has been reserved by BAE Systems for protection by patent. It is detailed in BAE Systems patent application 1107160.2.

2.3.3 Detectability

This task has also been explored in the recent ASTRAEA project. The project has produced a tool that assists engineers in deciding what sensors are needed on a system and what observations need to be made in order to be able to detect the occurrence of all possible component failures in that system. This enables efficient sensor selection and observation choice. This is done in consultation with the engineers, who know more about the reliability and availability of sensor than is possible to express in a model-based system. The work has the potential to be extended to make the detectability tool into a more general diagnosability tool. The work is detailed in [Snooke and Price, 2011].

2.3.4 Virtual prototyping

Before tools such as AutoSteve were available, once a circuit design had been drawn, it would be made into a physical prototype. Within Ford, this is known as yellowboarding, because all of the wires and components would be pegged out

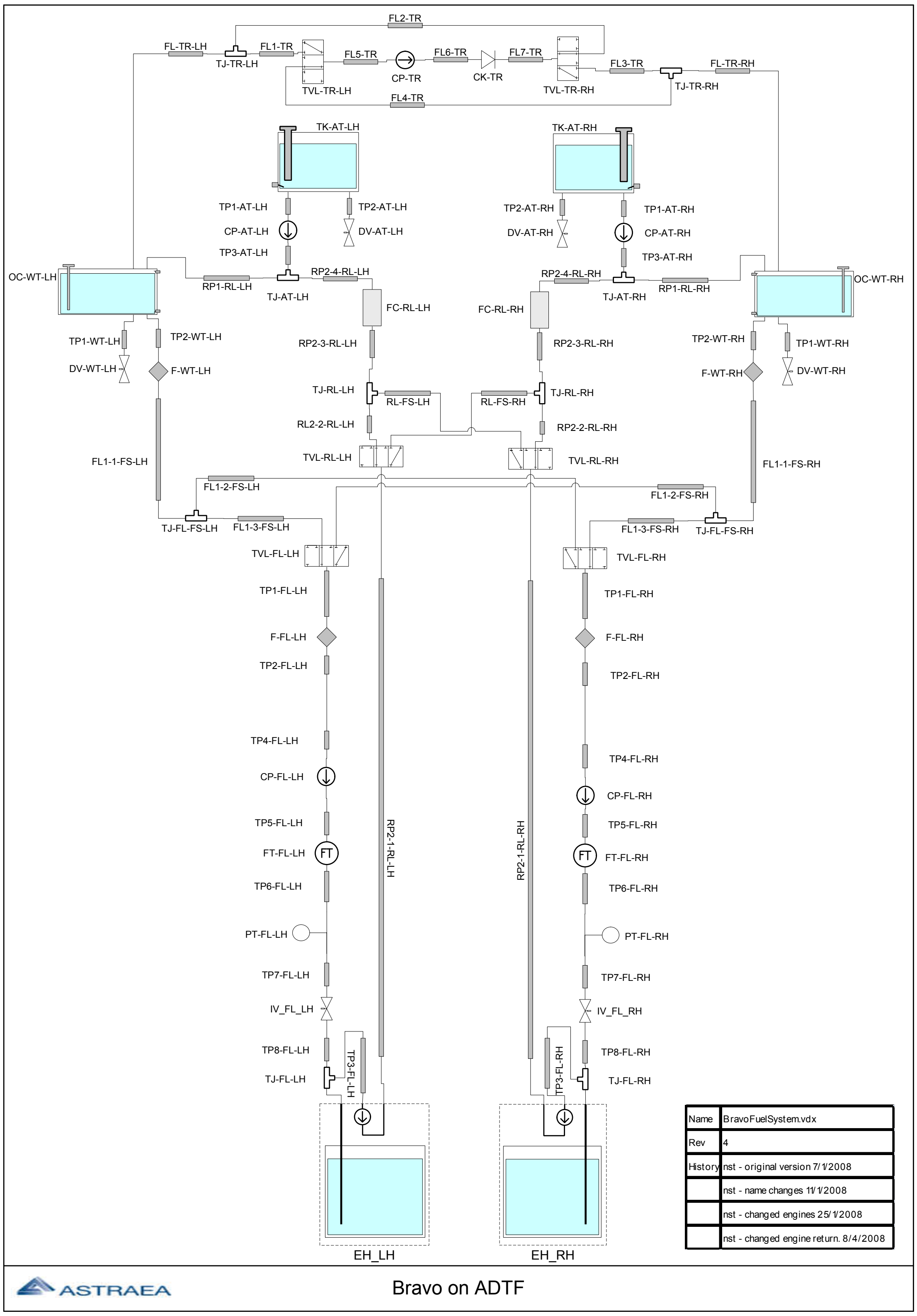


Figure 2.3: Fuel system diagram

on a large yellow board. The functions of the design would then be explored by a planned set of tests which turned switches and senses in the design on and off. The existence of a system such as AutoSteve means that most of this work is now unnecessary. The functionality can be explored on a computer by linking the simulation described here to the circuit design in the CAD tool used by the engineer. The engineers can go through their set of tests that would have been done with the physical yellowboard much more efficiently with the online simulation.

2.3.5 Sneak circuit analysis

Sneak circuit analysis detects the activation of unexpected functionality within a system. Typically, this occurs because current in a circuit takes an unexpected path through the circuit to activate that functionality. A classic example of a sneak circuit is given in the paper by [Savakoor, 1993]. This paper details a problem on a Boeing plane, as illustrated in Figure 2.4.

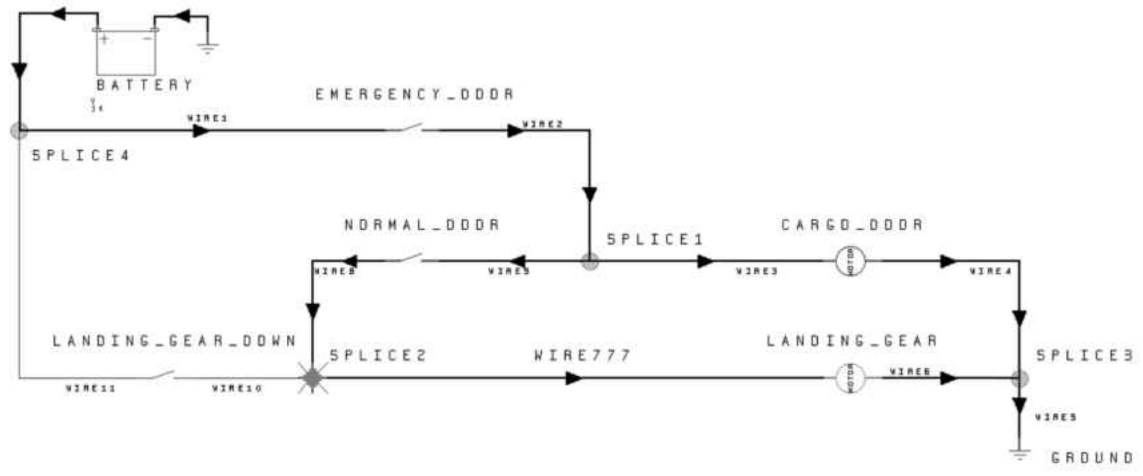


Figure 2.4: Aircraft sneak circuit

The cargo door and landing gear are supposed to be switched separately, with an override on the cargo door to open it in an emergency. In fact, because of the network that is the wiring in a modern vehicle, there is an unexpected path when the emergency switch is engaged, and activating the emergency override on the cargo doors also causes the landing gear to be activated.

The automated sneak circuit analysis tool [Price et al., 1996] works by declaring the expected functionality of an electrical system, and then exercising all possible functionality. The engineer declares the switch and sensor conditions for which a function should be activated. If there are any other sets of switch and sensor combinations that activated the functionality, then they are due to a sneak circuit.

The sneak circuit tool has been tested on many of the classic sneak circuit examples, and has a good track record of finding sneak problems, while not erroneously flagging up correct behaviour as sneak circuits (a problem with other automated sneak circuit tools).

2.4 Qualitative Circuit Analysis

Section 2.2 discussed the different levels making up the AutoSteve FMEA system. The lowest of these levels was made up of a qualitative circuit analyzer. This level is given a network of resistances by the component level (QCAT), and has to decide where current is flowing in the network, and in which direction.

This section discusses the different solutions that have been used for providing a qualitative circuit analyser. In several cases, the different techniques have been brought together. So, for example, later versions of CIRQ use series-parallel reduction to improve their performance.

2.4.1 CIRQ

CIRQ was the solution for providing qualitative circuit analysis for AutoSteve even before it was called AutoSteve. In the early research prototypes build for the Ford Motor Company [Hunt et al., 1993], when the FMEA software was still known as FLAME, the qualitative circuit analysis was done by CIRQ.

CIRQ is a shortest path algorithm for qualitatively solving electrical circuits, and was initially developed in the Computer Science Department in Aberystwyth University, and presented in [Lee and Ormsby, 1992] and [Lee and Ormsby, 1993] as pathfinding algorithms based in part on Dijkstra's shortest distance algorithm [Dijkstra, 1959].

CIRQ uses three levels of resistance, *Zero, Load* and *Infinite* and uses pathfinding algorithms to go in the forward (from power source to ground) and reverse (from ground to power source) directions. Each node in the graph representing the circuit is labelled with its Forward and Reverse values (hereafter referred to as **F/R**). These **F/R** labels give the number of nodes between the power source (forward) and the ground (reverse) and the node in question.

The original CIRQ algorithm was used as the central circuit solver for the FLAME software [Price et al., 1997] and the AutoSteve software [Price, 2000].

The performance of CIRQ was further developed through the identification *supernodes* and *Series-Parallel reducibility* within circuits [Lee, 1999]. The development of supernodes was based on work on graph-theoretic methods, using rewriting and replacement rules [Arnborg et al., 1991]. The series and parallel reduction rules were adapted from the work of [Mauss and Neumann, 1996] which used the transforms to reduce a circuit down to a single equivalent resistance value, and is described in section 2.4.3.

It is worth noting that the original CIRQ algorithm does not label the direction of current flow in wires. It identifies which parts of the circuit are active. Current flow in some wires can then be deduced because they are connected to source or sink. From there, flow can be propagated into other wires. Over time, the practical version of the CIRQ software acquired empirical rules that allowed more and more circuits to have current flow correctly labeled, but it was somewhat of an ad hoc process. The work described in this thesis enables a much more reliable labelling of flow in a circuit in many cases.

2.4.2 Early work at the Technical University of Munich

While reasoning about circuits has a comparatively long history in qualitative reasoning [de Kleer, 1984], there have only really been two groups doing serious research into qualitative modelling of automotive electrical systems: the Advanced Reasoning Group at Aberystwyth, and the Qualitative Reasoning Research Group at the Technical University of Munich (TUM).

Early work at TUM [Struss et al., 1995] attempted to build component based models similar to those developed by de Kleer and Brown [de Kleer and Brown, 1984] for fluid flow.

Connectivity Rules were given for the behaviour of each component. For example, the rule for a wire's connectivity was:

$$T_i.con - x - in = T_j.con - x - out \tag{2.1}$$

The rule for a splice was more complex, as you needed to worry about connectivity down two different paths:

$$T_i.con - x - in = min(T_j.con - x - out, T_k.con - x - out) \tag{2.2}$$

Given connectivity through a wire, the researchers at TUM attempted to apply rules for the current flowing through the wire;

$$T_1.current \oplus T_2.current = 0 \tag{2.3}$$

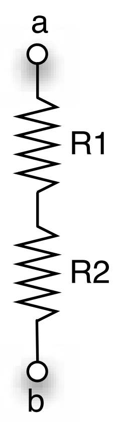
$$T_1.voltage = T_2.voltage \tag{2.4}$$

These rules worked fairly well for small circuits, but became less useful when applied to more complex automotive circuits. The TUM researchers rapidly abandoned this approach, and moved to using the series-parallel reduction strategy to reduce complex circuits to a more manageable size.

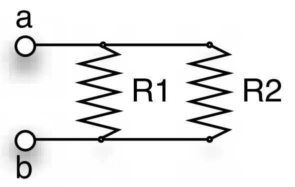
2.4.3 Series-Parallel reduction

This work was orginally applied to qualitative reasoning about electrical circuits by the Qualitative Reasoning Research Group at the Technical University of Munich [Mauss and Neumann, 1996, Struss and Price, 2003].

Series-Parallel reduction is an analysis technique used with electrical circuits that allows the simplification of a circuit by replacing pairs of resistances with a single resistance. The essential insight is that when resistors are in parallel or in series, they can be collapsed into a single resistance. When doing qualitative reasoning where the only resistance levels are zero, Load and ∞, then the load of the combined resistance is easy to calculate (see table 2.4.3).



(a) Resistive series circuit



(b) Resistive parallel circuit

For two resistances R_1 and R_2, the total resistance of the resistive series circuit of Figure 2.5(a) is found by the following equation:

$$R_T = R_{ab} = R1 + R2 \tag{2.5}$$

where R_T is total resistance of the circuit, and R_{ab} is resistance between the nodes a and b

The total resistance of the resistive parallel circuit of Figure 2.5(b) is found by the following equation:

$$\frac{1}{R_T} = \frac{1}{R_{ab}} = \frac{1}{R1} + \frac{1}{R2} \tag{2.6}$$

The introduction of the Series-Parallel (hereafter known as **S-P**) reduction transformations into the qualitative circuit solvers resulted in a decrease in execution time for three valued circuits. A three valued circuit is one that has three possible qualitative values for all the resistances in the circuit. The values possible are **Zero, Load** and **Infinite**, usually written 0, L, ∞. These values are ordered such that $0 < \text{L} < \infty$.

S-P reduction was initially done to simplify the circuit to make it easier to solve using a pathfinder. When replacing of a pair of resistance values is done in order to collapse a circuit, it depends on the structure of the two edges. If the two edges A and B are in series, then the sum of the resistances, or qualitatively, the maximum is taken. If the edges A and B are in parallel, then the equivalent replacement edge is the minimum of the two edges.

A	B	A & B in Series	A & B in Parallel
∞	∞	∞	∞
∞	Load	∞	Load
∞	0	∞	0
Load	∞	∞	Load
Load	Load	Load	Load
Load	0	Load	0
0	∞	∞	0
0	Load	Load	0
0	0	0	0
Equivalent		MAX	MIN

Table 2.1: Qualitative reduction rules for 3 valued electrical resistance

2.4.4 MCIRQ

The motivation behind MCIRQ was that the three levels of qualitative resistance handled by CIRQ (zero, Load and ∞) were not sufficient to be able to characterise the behaviour of many automotive systems. For example, a trickle current through a motor might be enough to provide an information signal, without activating the device. This type of strategy is often used by automotive engineers, and so ideally, a simulator would be able to differentiate between an activation current and a trickle current.

MCIRQ [Lee et al., 2001] is the Multiple resistance level Qualitative Circuit analysis tool. Instead of three levels of resistance (zero, Load and ∞), MCIRQ is able to work with a wider range of qualitative levels. While the theory of MCIRQ allows for n levels of resistance, in practice, five or seven levels of resistance are sufficient for most systems. For a five level system, the qualitative resistance levels would be zero, low, medium, high and ∞. The engineers are often able to specify these levels early in the design process, and so MCIRQ is able to do analysis of circuits almost as early in the design as CIRQ can.

The presence of these five levels enables visualization of of the different levels of activity occurring within the circuit. For a standard vehicle with a 12V battery system, three levels of activity can be shown in a circuit. These levels correspond to information flow (e.g. signal current to a CPU), activation flow (e.g. the current needed for activating a relay) and power level flow (the current needed to power a windscreen wiper).

In order to work, there must be an order of magnitude relationship between the different levels (so that adding resistances together does not take you to the next level of resistance). For a five-level quantity space, the relationship between the values zero, low, medium, high and ∞ would need to be:

$$zero < low < medium < high < \infty \tag{2.7}$$

MCIRQ works in the same way as the original CIRQ, by propagating F/R values through the circuit, but first of all, it uses **S-P** reduction to simplify the circuit as much as possible. The original intention of **S-P** reduction was that it would reduce a circuit down to so few nodes that qualitative analysis of the circuit would be trivial. That is not always the case, as some circuits cannot be reduced so easily. However, the technique can often simplify a circuit to such an extent that it makes CIRQ type analysis easier to perform, and so [Lee, 1999] integrates **S-P** reduction into the way in which MCIRQ works.

The outcome of MCIRQ was a more efficient electrical qualitative reasoning tool, able to analyse some circuits better than the previous CIRQ tool, and providing more accurate results.

However, there were still some outstanding challenges. In particular, [Lee et al., 2001] identifies the labelling of current flow direction and the resolution of bridge circuits as being *more* of a problem in MCIRQ than in CIRQ. [Lee, 1999] and an internal memo by Mark Lee (included in this thesis as Appendix A), set out those problems and potential solutions. The internal memo highlighted some circuits such as Figure 2.5 as being impossible to resolve in either CIRQ or MCIRQ.

This thesis has tackled several of the proposed strategies for improvement of the performance of MCIRQ. Professor Lee's main strategies for potential improvement were:

- Improvements to **S-P** reduction. These are discussed in detail in section 3.2, and have been implemented as part of the work of this thesis.
- Expansion of **S-P** reduced circuit back into original circuit showing flow.This is also discussed in detail in section 3.2, and has been implemented as part of the work of this thesis.

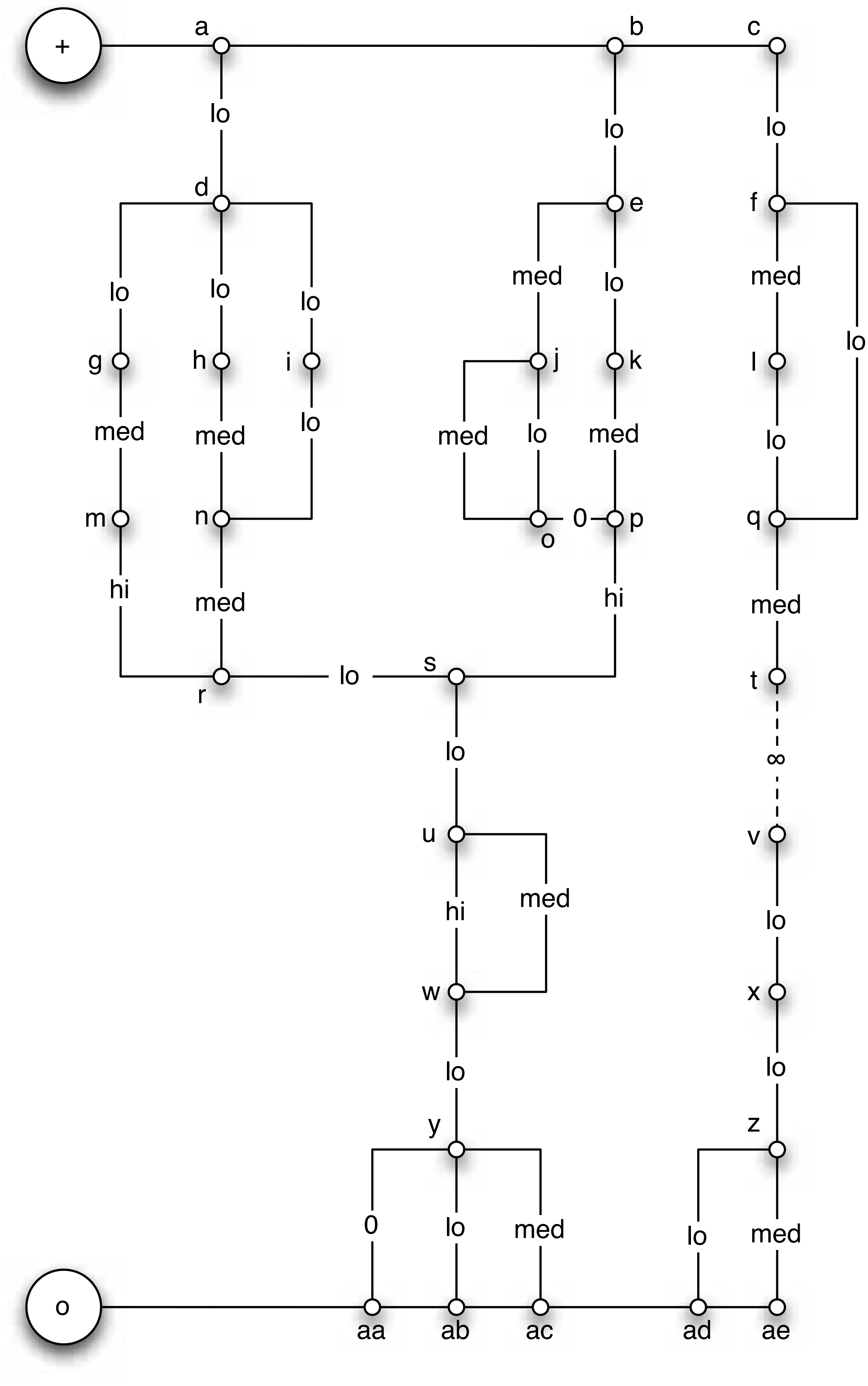


Figure 2.5: Complex circuit impossible to resolve in CIRQ or MCIRQ

- Implementation of supernodes in the segment table. This has been implemented in the new circuit simulator, and is discussed in section 3.2.
- Addition of star-delta transforms. These transforms assist with current flow assignment for circuits that do not reduce using **S-P** reduction. These have been added to the simulator implemented as part of this thesis since the completion of the original work detailed here.

Chapter 3

Improving Qualitative Simulation for Electrical Circuits

This chapter gives details of how performance of the electrical qualitative simulators CIRQ and MCIRQ can be improved by additional analysis. It proposes improvements to the series-parallel reduction analysis, and a much more principled, consistent approach to calculation of current flow.

The general strategy for these improvements comes from Prof. Mark Lee's technical report [Lee, 2002], included as Appendix A of this thesis, and from [Lee et al., 2001].

3.1 Performing Series-Parallel Reduction

3.1.1 Series-Parallel reduction

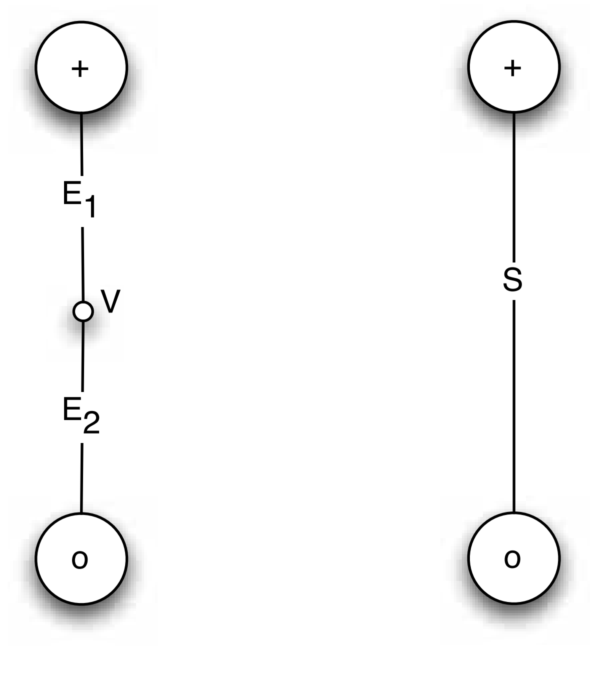
An electrical circuit can at steady state be represented by a network of resistances. This is even true of circuits containing capacitive and inductive elements as their resistance changes with respect to the stored energy of the device, but at any instant the component has a specific resistance. That network can be simplified by replacing sets of resistances in parallel or in series by a single resistance.

The method for doing this, using the theory given in section 2.4.3 is as follows:

3. Attempt to apply series reduction across the whole circuit until no more series reductions can take place.
4. Attempt to apply parallel reduction across the whole circuit until no more parallel reductions can take place.
5. If step 2 succeeded in performing parallel reduction, go back to step 1.

3.1.2 Series reduction

An edge E_1 and an edge E_2 are series reducible to a single edge S iff they share a common vertex V and that V has only two connected edges, being E_1 and E_2 and vertex V is not the supply or sink vertex.



(a) Series circuit before reduction

(b) Series circuit after reduction

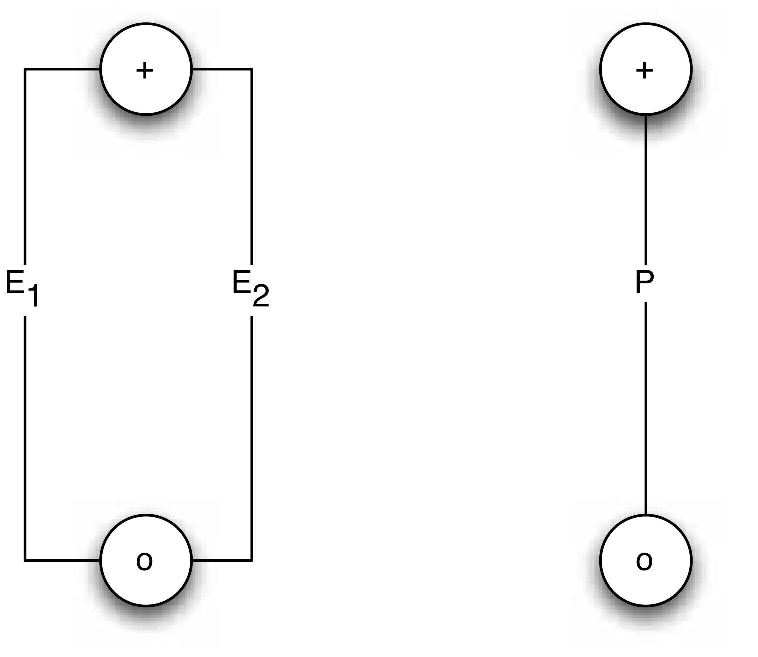
Figure 3.1: Simple example of series reduction in a circuit

Given an edge E_1 and an edge S which subsumes E_1 in series with another edge E_2. The resistance $R()$ of the edge S can be given as:

$$R(S) = max(R(E_1), R(E_2)) \tag{3.1}$$

3.1.3 Parallel reduction

An edge E_1 and an edge E_2 are parallel reducible to a single edge P iff they share two common vertices V_1 and V_2.



(a) Parallel circuit before reduction

(b) Parallel circuit after reduction

Figure 3.2: Simple example of parallel reduction in a circuit

Given an edge E_1 and an edge P which subsumes E_1 in parallel with another edge E_2. The resistance $R()$ of the edge P can be given as:

$$R(P) = min(R(E_1), R(E_2)) \tag{3.2}$$

3.2 Improvements to Series-Parallel Reduction

Professor Lee's paper in Appendix A describes the use of series-parallel reduction, intending it to be used in principled current flow analysis (see later). However, while implementing it as part of this thesis, a further set of drawbacks were discovered. The theory of Series-Parallel reduction described above works on well-structured circuits, but the application of this simulation to FMEA means that all sorts of unexpected circuits are generated for analysis. This has meant that circuits are often encountered with a number of complications that the basic **S-P** reduction cannot cope with.

The algorithm has the potential to solve these by being extended to deal with these special cases. Because of this, part of this thesis has included devising ways of dealing with the following types of special case:

- Floating Edges - an edge E that provides the only connection to a vertex V;
- Loops - an edge E that connects to a vertex V on both ends;
- Infinite Edges - an edge E whose resistance is infinite;
- Zero Edges - an edge E whose resistance is zero.

3.2.1 Special Cases

This section considers how **S-P** reduction can be extended to deal with each of these cases that is ignored by the simple **S-P** reduction theory.

3.2.1.1 Floating Edges

A floating edge is an edge that provides the only connection to a single vertex. The edge E1 in Figure 3.3 shows this diagrammatically.

In the type of single shot, direct current, simulation that is being performed by this algorithm, floating edges can be removed as they do not provide any electrical connectivity to any other part of the circuit. They connect to a vertex, providing the only connection to that vertex. That vertex will be held at an electrical potential, but no current will flow through the edge.

Edges of this type can be identified by processing all the edges in the circuit. Each end of the edge is examined to see if the vertex it is connected to is not the source or ground vertex. As long as it is not the source or ground vertex, if the number of edges connected to that vertex is one, then that edge can be removed from the circuit. The edge is labelled as being floating for current resolution later.

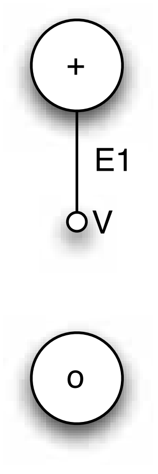


Figure 3.3: A floating edge

3.2.1.2 Loops

A loop is an edge that connects to a single vertex on both ends. The edge E2 in Figure 3.4 shows this.

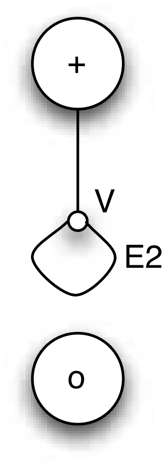


Figure 3.4: A loop edge

This type of edge has both its ends held at the same potential, and therefore, no current will flow through the edge. The loop contains nothing but a single edge that represents a resistance, so there is no possibility of the edge doing anything "useful".

Edges of this type can be identified by processing all the edges in the circuit. Each end of the edge is examined to see if the vertex connected to one end of the edge is the same vertex connected to the other end of the edge. If the vertices are the same then the edge can be removed from the circuit. The edge is labelled as being a loop edge for the current determination stage.

3.2.1.3 Infinite Edges

An infinite edge is one whose resistance is infinite. The edge E3 in Figure 3.5 shows this.

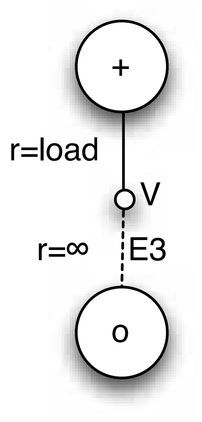


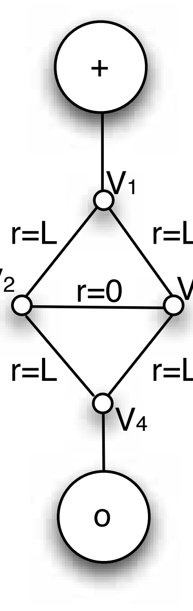
Figure 3.5: An infinite edge

This type of edge cannot carry any current. It may have been added to the structure of the circuit representing an open switch or some other similar construct. During simulation, all the edges can be examined in turn and if an edge's resistance is infinite, then the edge can be removed from the circuit. An infinite edge can never carry any current, so this is noted for the current determination stage.

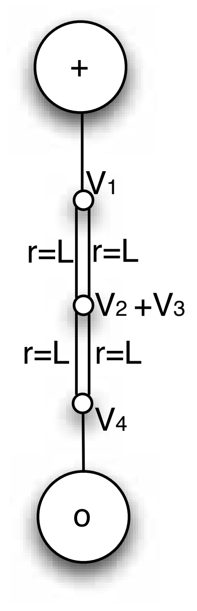
3.2.1.4 Zero Resistance Edges

A zero resistance edge can be removed from a circuit, and the two vertices that it connects can be combined into a single vertex. This can help simplify some otherwise non-reducible circuits by altering the topology. Figure 3.6 gives two examples of how the collapse of a zero resistance edge can allow the S-P reduction to continue. Both of the circuits in Figure 3.6(b) and 3.6(d) are immediately parallel reducible allowing simplification to continue.

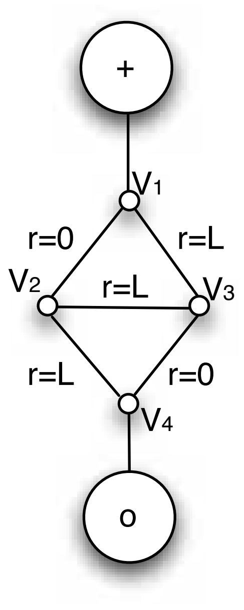
It should be noted that this type of circuit topology does not normally occur in electrical circuits during normal operation, however they do quite occur when



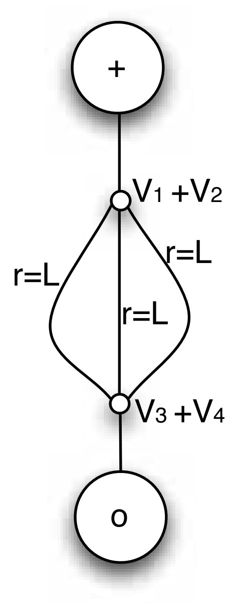
(a) Non-reducible circuit



(b) Reducible circuit equivalent to 3.6(a)



(c) Non-reducible circuit



(d) Reducible circuit equivalent to 3.6(c)

Figure 3.6: Zero resistance example circuits

faults such as short circuits are introduced. As one of the main reasons for developing this new circuit simulator is for use with FMEA, these type of short circuit faults will be required to be simulated often, making this functionality essential.

Current determination for circuits simplified in this way is a little more complicated than for other cases, as in order to determine current direction through a zero resistance edge we have to look at the other current flows into and out of the vertices attached to the ends of that edge.

If the current flows that are known are both directed into the vertex in question, then the current flow that is unknown must be directed out of the vertex, and vice versa. If there are two current flows known, one into the vertex and one out from the vertex, then the magnitude of the current flows must be examined, or the current flows at the vertex at the other end of the connection must be examined.

3.2.2 Enhanced Series-Parallel Reduction Algorithm

The main **S-P** reduction loop becomes a sequence within a loop whereby the algorithm only proceeds to the next step if the previous steps have not changed the structure of the circuit being considered. As soon as one of the steps causes a change to the circuit structure, execution of the loop starts again with the first step.

Before the loop begins we can go through the graph and remove all the Infinite Edges. If an edge has its resistance changed during a run, then the graph gets re-generated, because a change of resistance will require the reduction process to be re-run.

The steps in the loop are as follows:

1. Remove Floating Edges
2. Remove Loop Edges
3. Perform Series Reduction
4. Perform Parallel Reduction
5. Collapse Zero Edges

Treatment of infinite edges, floating edges and loop edges can all be done in very similar ways, as they all require removal of the edge in question. In none of these cases will current be flowing across the node, and therefore there is no need to calculate current direction for the node.

Once any floating, loop or infinite edges have been removed from the circuit, then a series reduction of the circuit is performed, and if any pair of edges is reduced to a single edge, an attempt must be made to remove floating edges again.

An attempt to collapse zero resistance edges is only made once Series and Parallel reductions have failed to achieve anything. When collapsing a zero resistance edge E, we combine the two vertices $V1$ and $V2$ on either end of the edge E into a new single vertex $V3$.

3.2.3 Current flow determination

A circuit that has had a full **S-P** reduction done on it should result in a single resistance joining power and ground. The size of this single resistance determines the maximum amount of current that can flow through the circuit.

It should be noted that the result of this thesis is only the central core of a simulation system, therefore it is assumed that there will be a program wrapping this engine. The complete system that has been used on the ASTRAEA project that was developed at Aberystwyth, shown in Figure 3.7 wraps this central simulation engine with several layers of reasoning at component level, circuit level, and system level. These different layers add behaviour and function to the system as well as adding the facility to be able to simulate circuits of different type that interact with each other (an electrical system might operate valves in a fluid flow system).

Current or flow is only determined for resistances when they are requested by the calling program. This is done to reduce the amount of unnecessary computation that may have to be done. When current is requested for a top-level edge, the program recursively checks if the edge has been subsumed by another during the **S-P** reduction phase. At the end of this process if we arrive at a single resistance, then we can unwind the recursion applying current determination rules as we go depending on whether each reduction step was a series or parallel step.

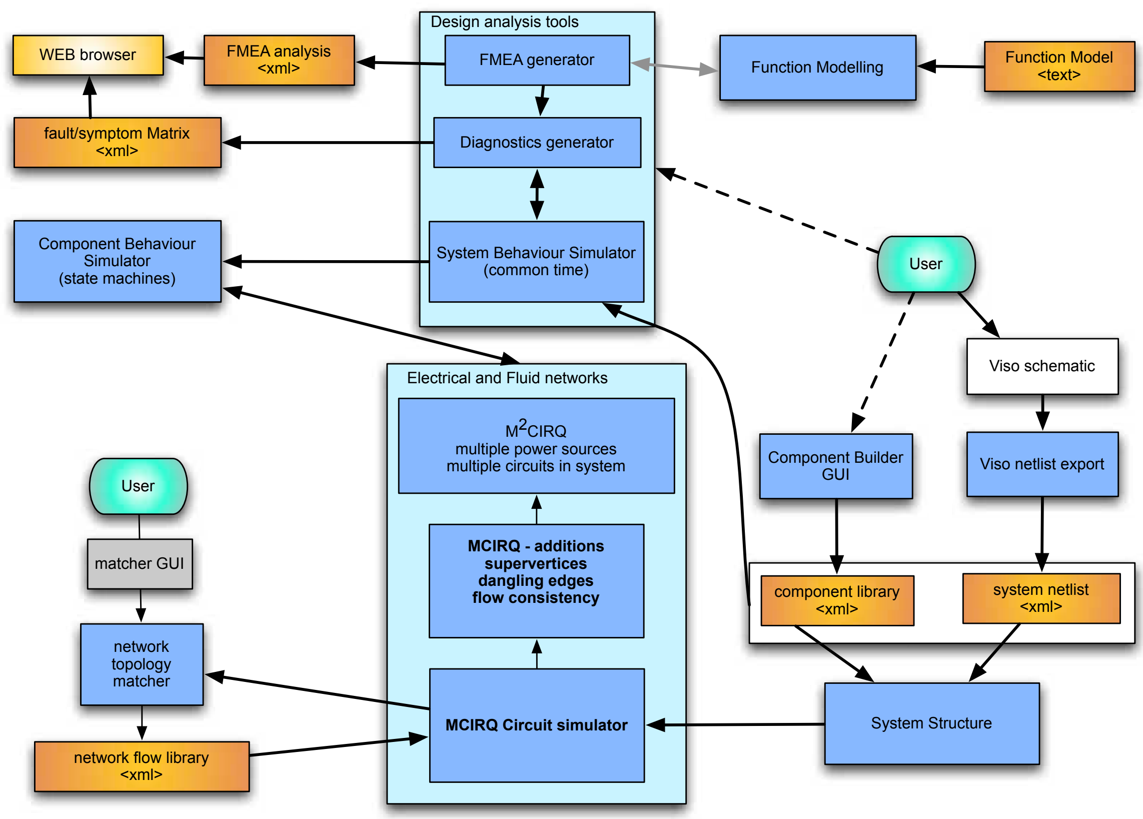


Figure 3.7: Overview of ASTRAEA software system

Short circuit determination has to be done, because if the branch that we are interested in is in parallel with another branch of Zero resistance, then all the current will be carried on the Zero resistance branch. The determination of short circuiting is done during parallel phase of the **S-P** reduction, and sets a flag in the short circuited edge. If an edge E_1 has been short circuited by a parallel edge E_2 with a resistance of Zero, then edge E_1 will pass no current.

Current determination for a single resistance is done through a simple look-up into the quantity space, which determines how currents map onto resistances. The default 5 value quantity space has values as shown in table 3.1. Note that the maximum current that can be achieved is labelled as *Max* rather than ∞ as non-idealised power sources cannot supply infinite current, and also that there can be resistances that appear infinite to non-idealised power sources.

Resistance	Zero	Lo	Med	Hi	Inf
Current	Max	I+	I	i-	0

Table 3.1: Default quantity space for resistance and current, and mapping between them.

In the case of a series step, the current passing through the edge is the same as the current passing through the edge that subsumed it.

Given an edge E_1 and an edge S which subsumes E_1 in series with another edge E_2. The current $I()$ of the edge E_1 can be given as:

$$I(E_1) = I(S) \tag{3.3}$$

Current determination for a parallel step is somewhat more complicated. If the resistance of the edge E_1 is the same as the resistance of the edge P, then the current $I()$ of the edge E_1 is:

$$I(E_1) = I(P) \tag{3.4}$$

If the resistance for the edge E_1 is greater than the resistance of the combined edge P, then it follows that:

$$I(E_2) = I(P) \tag{3.5}$$

and

$$I(E_1) < I(P) \tag{3.6}$$

In actual fact, $I(E_1)$ is the next qualitatively lower level of current than $I(P)$.

In the three valued version of CIRQ, the current assigned to $I(E_1)$ would be equal to $I(P)$, which would provide results which were pessimistic, always reporting the maximum current or over the maximum current that can occur, which was acceptable as the simulations were being used to produce Failure Mode Effects Analysis reports, therefore generating results that provide a good margin for safety.

A more accurate solution is to generate new current levels in the quantity space. In order to do this we have to introduce new qualitative values into the quantity space that we use to measure current flow.

3.2.3.1 Qualitative Value Generation

A problem occurs where $I(P)$ is the lowest level of current above zero, in which case we have to introduce a new qualitative value into our quantity space to indicate that some current is flowing, but it is less current than would flow through the highest value resistor in the quantity space if that resistor was directly connected to power and ground. The circuit in Figure 3.8 has an example of a number of parallel resistances that will demonstrate this functionality.

On the first occasion that we encounter such a section of circuit, the system will introduce a new level of current flow. That flow is automatically named by appending a $-$ sign to the end of the lowest flow value in use. On subsequent occasions that this situation is encountered, the value already existing in the quantity space is used, and another level is only introduced if a new level is necessary.

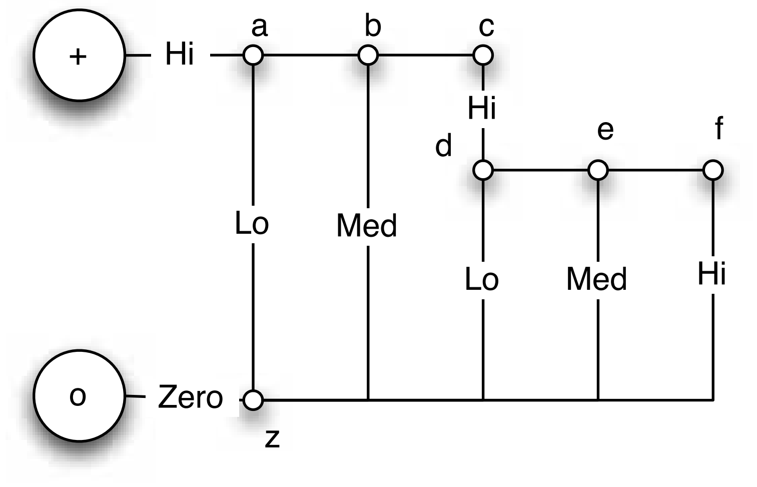


Figure 3.8: Parallel circuit to demonstrate qualitative value generation

Chapter 4

Implementation

4.1 Design decisions

The data structures to be represented were examined, and outline designs were produced. These were used to help determine a programming paradigm for the implementation of the software, and hence a programming language.

The object-oriented paradigm was chosen, with the Java programming language as the implementation language. This decision was influenced by other software that already existed that the new network solver was to integrate with. The existing software was already implemented in Java, and whilst another language could have been chosen, it was felt that it would be more convenient to keep a single language throughout the system. Java version 5 was used, as the development is primarily being done on a Macintosh platform, and the Java 6 environment is not yet complete or mature enough to rely on for this project. Having said that, most of the code is written such that it will run on Java 6.

The development environment chosen was the Eclipse IDE, synchronising with a CVS server back-end run on a separate machine to ensure change and version control throughout the project. This may only be a single developer project, but that is no reason to sacrifice good software development practices, and the code was check-pointed after each and every change, such that rollbacks can take place in a controlled fashion. The Eclipse IDE makes this straightforward, and therefore there was even less reason not to do it.

In his internal memo on MCIRQ [Lee, 2002] (Appendix 1), Mark Lee suggests the use of a segment table to store the data on each series or parallel branch both before and after compression. This method might be suitable for a non O-O language, but having chosen an O-O paradigm, it was felt that it would be more natural to use objects to represent the edges and vertices, with an edge knowing whether it had been compressed. If an edge knows that it has been compressed, then the edge would also know which edge it had been compressed into, removing the need for the segment table. A segment line and segment table object were used to encapsulate the series and parallel reduction algorithms.

In an Object-Oriented paradigm, the circuit knows about the edges and vertices that comprised it, and is able to perform certain actions on those vertices and edges. The higher level program does not have to know very much about the existence of the Edges and Vertices, as long as relevant actions could be performed on parts of the circuit.

4.2 Implementation Process

The code was developed using a test-driven development methodology. Tests were written in advance of any major development, and were enhanced as features develop. The JUnit test suite was used to support the testing, and a number of *sanity checks* were introduced throughout the code, ensuring that parameters are within correct bounds, and verifying that the algorithm is behaving correctly. These may actually slow down the final algorithm, by performing redundant comparisons, but in the author's opinion they are worth putting in, as they cut down on debugging time. Some of these may be removed when the new simulator is integrated into the larger system to speed up the simulations, but they have been left in for the purposes of the thesis.

A number times during development, it was found necessary to re-factor the code to provide a class structure that had less duplication in the code base. This also allowed the code to implement an inheritance hierarchy and made the code more readable and maintainable. The re-factoring tools in the Eclipse IDE made this process a little easier, and helped expose some minor bugs that had not previously been detected in the code base.

The initial system as implemented would only do *single shot* simulation. The system would run a simulation on a static circuit with no resistance changes to establish current flow through each resistance in a stable state. The simulator was built into a system that altered resistance values between runs, then the simulation engine detected any changes in resistance and marked the circuit as "dirty", indicating that the circuit has changed in structure since the previous simulation had run and the results were not applicable to the altered circuit. The simulation core was integrated into the large system that has been developed for the ASTRAEA project at Aberystwyth. If the calling system attempts to query the current of an edge while the circuit is dirty, a simulation process is triggered before results are returned.

Another small limitation of the system as implemented initially was that current flow direction was not determined. This was a small oversight that was simply rectified by the addition of a small amount of code to allow the querying of flow direction. It was a very straightforward extension, as current direction is determined at the same time that current flow is determined. This can be seen in some of the results.

4.2.1 Structural representation

A graph representation, with Edges having knowledge of connected Vertices, and Vertices knowing about Edges. An Edge knows if it was created by an external call - from the main program defining the structure of the circuit. If it was not created by an external call, then it is an internal addition to the graph during the compression phase - these are edges that are added to replace a pair of series or parallel edges. An Edge also knows if it has been "subsumed into" another edge by this process, and knows which edge has subsumed it, so that during current determination, the network can be easily traced from the top level down to the lowest possible level of the collapsed circuit.

A *Circuit* has a Hashtable of all the *Vertices* indexed by name, and an ArrayList of all the *Edges*.

An *Edge* knows if it was created by an external call, or if it was created internally by the S-P reduction process.

A *Vertex* only knows about the current edges - ones that have not been subsumed by Series or Parallel reduction.

4.2.2 Series-Parallel reduction

The Series-Parallel reduction is implemented with the main interaction method in Circuit. This method *doSPReduction* calls series and then parallel reduction methods, working out if any reduction has been done at this step, and then repeating until no reductions were done in the cycle. There is a small amount of optimisation that does not run the method if a reduction has been done previously, unless structural changes have been made to the circuit.

4.2.3 Current determination

The system does not perform current expansion for the complete circuit, but only for queried Edges. This saves a lot of unnecessary labelling of Edges with current values that are not needed. Most of the time, only a few edges are queried in order to determine the functionality of the circuit. All that is required to determine the current of a particular Edge is to query down through the SubsumedBy hierarchy when the Edge is queried. The result may be able to be cached, as long as it is nullified when any structural changes are made to the circuit, as structural changes will invalidate any cached results. The results are not cached in the system as currently implemented, as it was not found necessary to improve the performance in the current environment

4.2.4 Non S-P reducible circuits

Circuits that are non **S-P** reducible have previously been treated with star-delta transforms. This proved to be difficult in qualitative circuits, and did not work for all cases. I present an alternative method of dealing with non **S-P** reducible circuits.

When a query is done on an edge that is on a non **S-P** reducible segment of circuit, the system then calls the registered StructureCallback. See Code Fragment 1 for implementation detail. This method provided by the calling program is required to return a Current value for the given edge in the provided circuit. An empty example call-back is shown in Code Fragment 2.

In order to implement this method, the programmer will probably need to query the user to assist in determining the current flow through the edge specified. Some user responses will be able to be cached, and a knowledge base could be built up enabling the calling program to fill in values without always having to interact with the user. This is something that will be developed more in the future, when the simulator is integrated with a larger system.

With the introduction of the super-nodes that aggregate vertices connected by zero resistance arcs together with the removal of hanging edges and loop edges, non S-P reducible circuits occur much less often than was originally anticipated. The occurrence of a non S-P reducible circuit more often indicates some form of error in the circuit design.

Code/Data fragment 1 The StructureCallback interface.

```
package uk.ac.aber.rcs.mcirq;

public interface StructureCallback {
   /**
    * This method is called when the internal resolver is not able
    * to determine current.
    * The user has to provide a method that does something with
    * the circuit (that contains the edge in question)
    * and return a value for the current of thisOne.
    * @param thisOne the edge to determine current for.
    * @param circuit the circuit containing thisOne
    * @return a current value for thisOne.
    */
   public Current resolveEdge(Edge thisOne, Circuit circuit);
}
```

Code/Data fragment 2 An example (empty) structure callback.

```
class CallbackTest implements StructureCallback {
   public Current resolveEdge(Edge thisOne, Circuit circuit) {
      System.out.println("Edge to be resolved: "+thisOne.getName());
      System.out.println((circuit.getSegment(thisOne)).getXML());
      return null;
   }
}
```

Chapter 5

Results

5.1 Series Parallel reduction

Code fragment 3 is an example of the XML type output produced after S-P reduction has been carried out on the main circuit in Figure 5.7, and is represented in a diagram in Figure 5.1. This output is taken just before the final Parallel reduction that results in a single resistance between the power and ground vertices. The detail of how this is achieved can be seen in the copy of [Lee, 2002] in Appendix A.

This illustrates the type of output that can be generated from the **S-P** reduction, and demonstrates that a circuit seemingly as complex as Figure 5.7 can be simply collapsed to a single resistance. The next step on from Figure 5.1 will be to reduce the two parallel resistances to a single resistance with a resistance of *med*. It should be noted that this was the result before the optimisations to remove infinite resistances were introduced.

5.2 Automotive electrical circuits

I will illustrate the usage of the circuit analysis tool by applying the tool to analyse a simple headlamps circuit from a car. This is a circuit that has been used to test tools in the past, and has proved useful to ground the examples. The circuit is

Code/Data fragment 3 Example XML output from doing S-P reduction on main example circuit

```
<CIRCUIT>
<QSPACE>
   <RVAL>Zero</RVAL>
   <RVAL>Lo</RVAL>
   <RVAL>Med</RVAL>
   <RVAL>Hi</RVAL>
   <RVAL>Inf</RVAL>
</QSPACE>
<STRUCTURE>
<EDGE NAME="zad=ad-|zae=ae-=xz=vx=qt=tv=+c=cf=fq|fl=lq">
   <FROM>-</FROM>
   <TO>+</TO>
   <RESISTANCE>Inf</RESISTANCE>
</EDGE>
<EDGE NAME="wy=su=uw|uw=yaa=aa-|yab=ab-|yaa=aa-|yac=ac-=rs=+a=ad=mr=dg=gm|
                       nr=dh=hn|di=in|ps=+b=be=ek=kp|op=ej=jo|jo">
   <FROM>-</FROM>
   <TO>+</TO>
   <RESISTANCE>Med</RESISTANCE>
</EDGE>
</STRUCTURE>
</CIRCUIT>
```

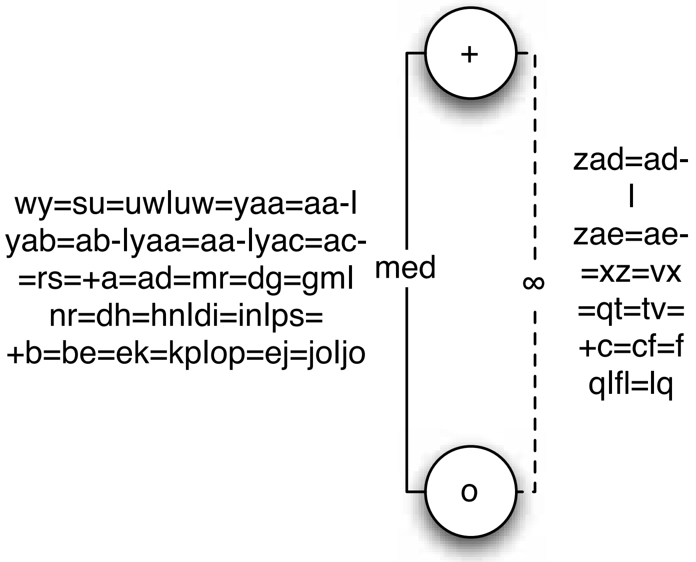

+
o
wy=su=uwluw=yaa=aa-l yab=ab-lyaa=aa-lyac=ac- =rs=+a=ad=mr=dg=gml nr=dh=hnldi=inlps= +b=be=ek=kplop=ej=joljo
med
∞
zad=ad- l zae=ae- =xz=vx =qt=tv= +c=cf=f qlfl=lq

Figure 5.1: Diagrammatic representation of the S-P reduced main example circuit.

shown in Figure 5.2.

This circuit has been simplified to two switches, a single-pole single-throw switch to apply power to the circuit, and a single-pole double-throw switch to change between dipped and high-beam lamps. More usually, these will actually be relays, operated by other switches, however the simplification is valid, as the switches in the circuit as presented may be regarded as the poles in the relay. This is because the simulator is not intended to address the behavioural aspects of the circuit - that is addressed at the QCAT and higher levels of the system.

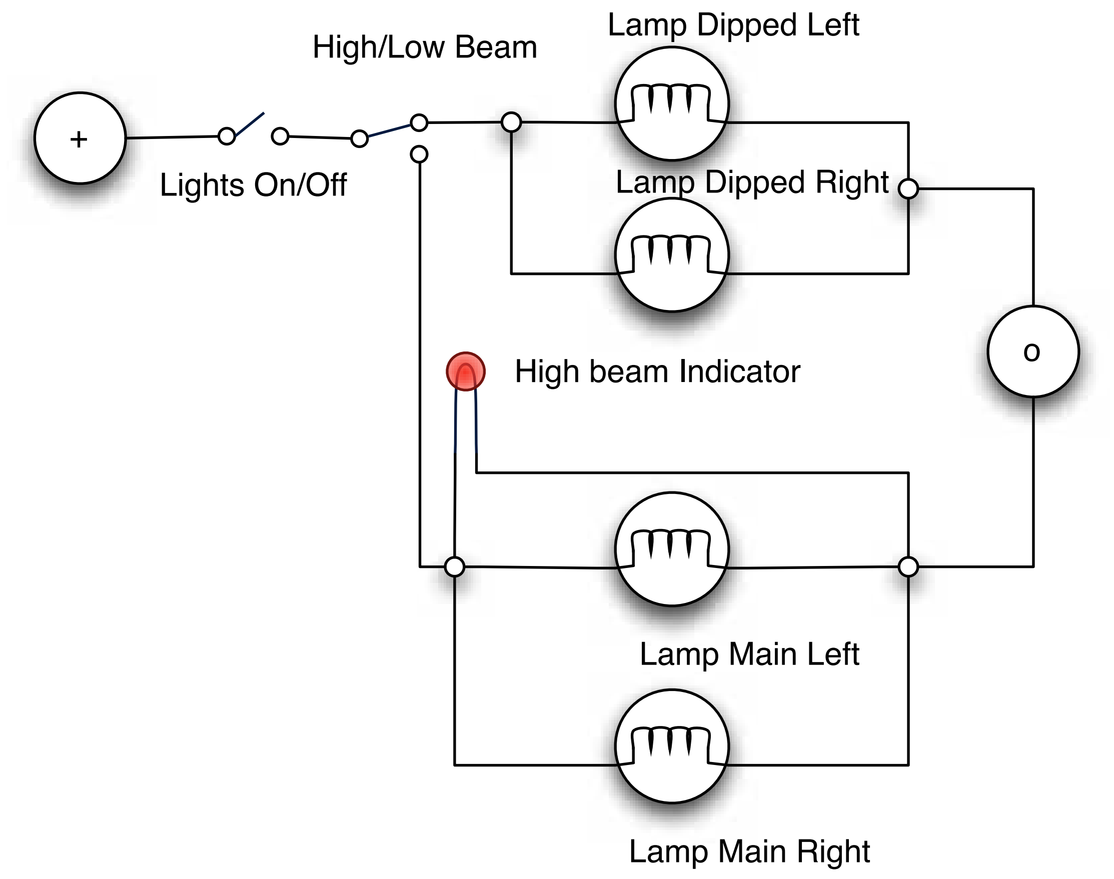


Figure 5.2: Simple headlamps circuit.

The circuit is represented by reducing it to a graph structure of vertices and edges. Some of the edges are named, as those are ones that we will be interested in either examining the current flowing or changing resistance. This process is usually done automatically by a net-lister program, generating the correct input data, or *net-list* from a CAD drawing tool. In this instance this has been done manually. This results in the graph shown in Figure 5.3. Resistance values have not been included on this diagram, but the vertices have been labelled appropriately. These labels would be generated by the net-lister.

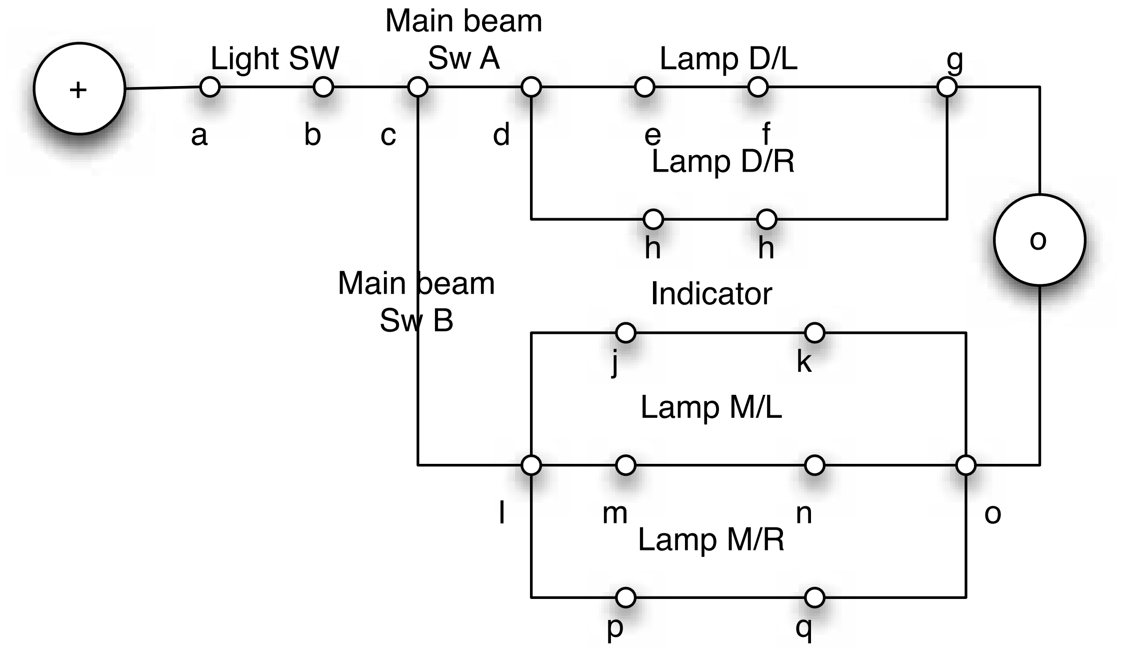


Figure 5.3: Simple headlamps circuit as vertices and edges for CIRQ.

This net-list is then translated into a data-set for the program, currently entered directly into a JUnit test file. The setup code for the circuit is given in Code fragment 4.

The switch positions are changed by calling methods in the circuit that set up the correct resistances. The simulation method is then called, which performs the S-P reductions. When the simulation completes we get a result for the current flowing through the edges that we are interested in.

The headlamps are turned on by changing the resistance of *Lightsw* to *Zero*, and the main beam is activated by changing the two resistances of the double throw switch. The single pole switch is modelled as two resistances, one of which has Zero resistance and one of which has Infinite resistance. The changing of resistance values is handled by QCAT, at a higher level, and is not relevant to discuss in any more detail here. The calls to make the resistance changes are shown in Code fragment 5.

The results of running the simulation are given in Code fragment 6, where it can be seen that there is current flowing through the Main beams of the headlamps (LampML and LampMR) and also through the Main beam indicator lamp. These

Code/Data fragment 4 Example JUnit test data set for headlamps circuit

```
c.makeEdge("+", "a", "Zero");
c.makeEdge("a", "b", "Inf", "Lightsw");
c.makeEdge("b", "c", "Zero");
c.makeEdge("c", "d", "Zero", "Dipswa");
c.makeEdge("d", "e", "Zero");
c.makeEdge("e", "f", "Med", "LampDL");
c.makeEdge("f", "g", "Zero");
c.makeEdge("d", "h", "Zero");
c.makeEdge("h", "i", "Med", "LampDR");
c.makeEdge("i", "g", "Zero");
c.makeEdge("g", "-", "Zero");
c.makeEdge("c", "l", "Inf", "Dipswb");
c.makeEdge("l", "m", "Zero");
c.makeEdge("m", "n", "Med", "LampML");
c.makeEdge("n", "o", "Zero");
c.makeEdge("l", "p", "Zero");
c.makeEdge("p", "q", "Med", "LampMR");
c.makeEdge("q", "o", "Zero");
c.makeEdge("l", "j", "Zero");
c.makeEdge("j", "k", "Hi", "Indicator");
c.makeEdge("k", "o", "Zero");
c.makeEdge("o", "-", "Zero");
```

Code/Data fragment 5 Changes to circuit structure

```
c.changeResistance("Lightsw", "Zero");
c.changeResistance("Dipswa", "Inf");
c.changeResistance("Dipswb", "Zero");
```

results when integrated into the tool would actually display on the circuit diagram, or could be used by the higher levels of the Design Analysis toolset.

Code/Data fragment 6 Results of simulating headlamps circuit and interrogating interesting edges.

```
LampDL 0 NONE
LampDR 0 NONE
LampML I FORWARD
LampMR I FORWARD
Indicator i- FORWARD
```

5.3 Current determination

Current determination is done as described in Section 3.2.3, where the example circuit is first presented in Figure 3.8

In Figure 5.4, the results of that simluation is shown, giving current values on the diagram. All the current values *"qualitatively smaller than" i-* are automatically generated qualitative values by the system.

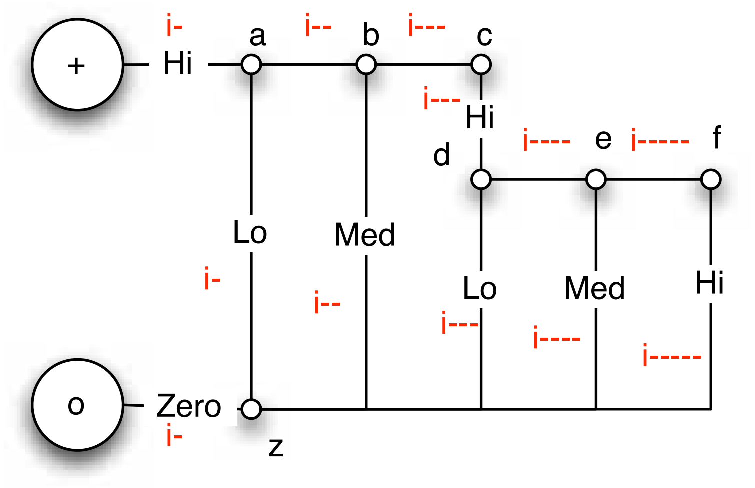
This circuit was deliberately designed to test the qualitative value generation code, and is not representative of any circuit that will occur in the real world. This circuit includes a more complex parallel topology than we have experienced in the automotive and aerospace work [Snooke, 2007]. The qualitative value generation only occurs in parallel circuits, so the test circuit was only needed to have multiple parallel paths.

The results shown in Data Fragment 7 are as expected from hand-working the algorithm, and appears to demonstrate that to at least 4 levels the current generation works predictably.

5.4 Experimental Validation

A number of test circuits are described in [Lee, 2002] and [Lee et al., 2001], and these were used as initial validation cases for the new code. These circuits have

Figure 5.4: Qualitative value generation circuit with flows labelled

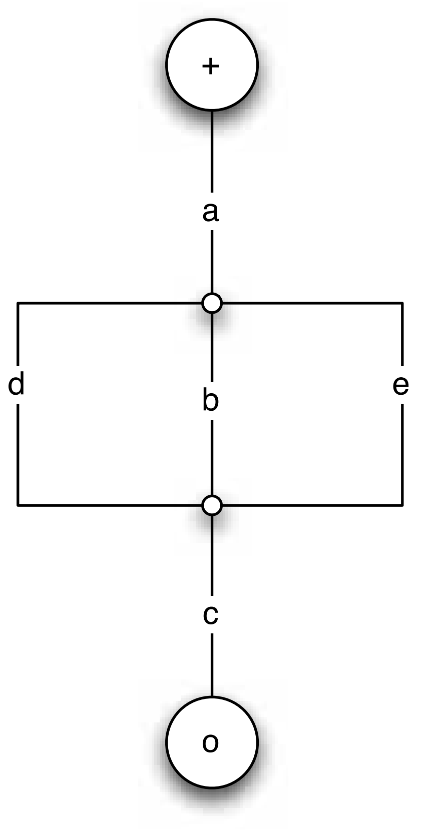


Code/Data fragment 7 Example output from test case for flow qualitative value generation

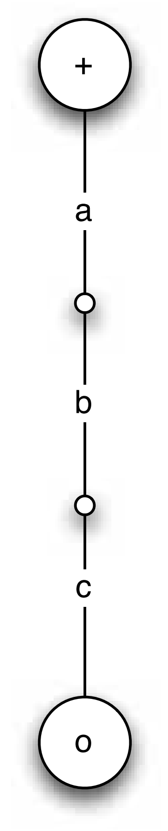
```
+a Hi Current = i-
ab Zero Current = i--
bc Zero Current = i---
cd Hi Current = i---
de Zero Current = i----
ef Zero Current = i-----
az Lo Current = i-
z- Zero Current = i-
bz Med Current = i--
dz Lo Current = i---
ez Med Current = i----
fz Hi Current = i-----
```

been used in the past to prove the correctness of implementations of CIRQ and MCIRQ, and are well understood.

The simple examples in Figure 5.5 are very basic test cases that provide some confidence that the system is working, and by substituting different values of resistance for *a, b, c, d* and *e*, quite a lot of functionality can be tested. If, for example, in Figure 5.5(a), one of the edges *b, d* or *e* has the value zero for resistance and the other two edges are non-zero, then it should be observed that the other two edges have no current flowing as they are shorted out by the zero resistance edge, resulting in the remaining circuit being treated as Figure 5.5(b). A comprehensive analysis of the system using this circuit is provided in section 5.4.1.



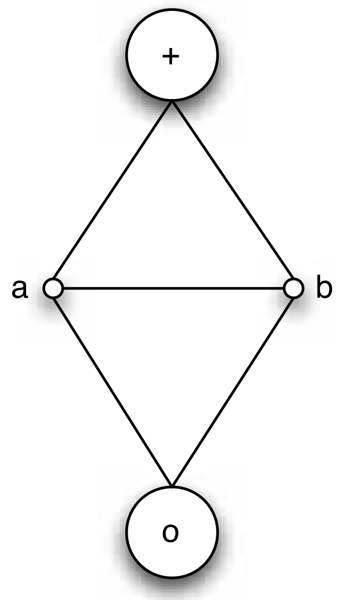
(a) Simple parallel circuit



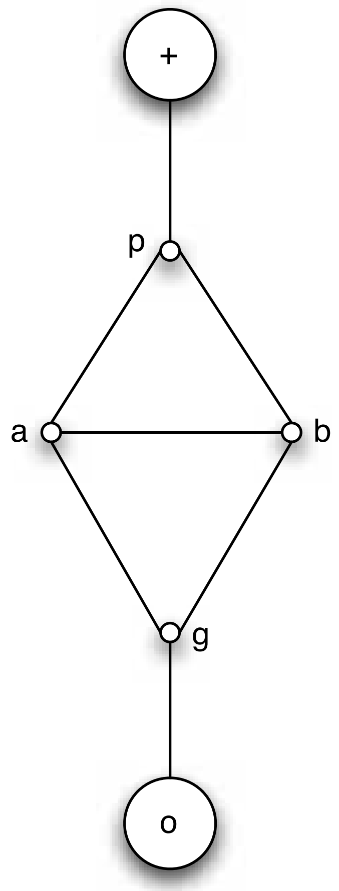
(b) Simple series circuit

Figure 5.5: Simple example circuits

The two circuits that encompass the main problems with the approach to collapsing the circuit are the simplest of bridge circuits as shown in Figure 5.6(a) and the simple double star, that occurs in Figure 5.6(b). These two circuits will cause problems as they are not Series-Parallel reducible. When a non **S-P** reducible circuit is detected, a function that is part of the higher level system, that is registered with the simulator is called with the collapsed version of the circuit so far. This call-back mechanism will operate in a way to allow simulation to proceed when the reduction process fails at some point. The circuits in Figure 5.6 is used to test that the call-back mechanism operates.



(a) Delta circuit



(b) Star circuit

Figure 5.6: Simple problematic circuits

Figure 5.7 is the primary example circuit given in [Lee, 2002], and was used as the main general test case. It demonstrates most of the main structural issues that can arise in a full Series-Parallel reducible circuit, including the new features of the addition of a new level of current in one of the circuit branches.

In this example circuit names are only assigned to the vertices, and not to the edges. When an edge is not assigned a name, it is given a name which is the aggregate of the two vertices that it joins. This can cause two edges to have the same name if they connect the same two vertices, but this does not affect the performance of the Series-Parallel reduction, as the program refers to each

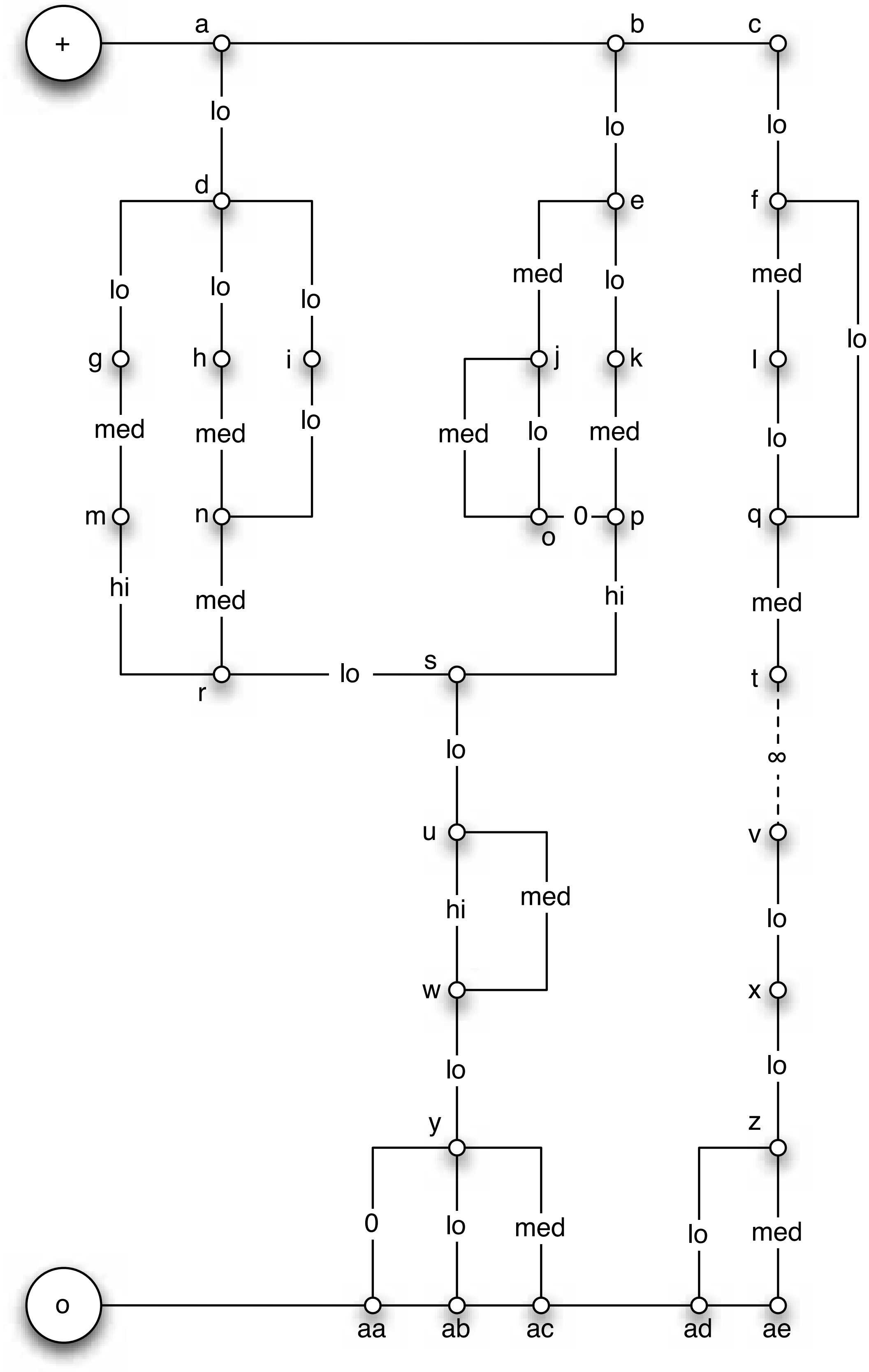


Figure 5.7: Main example circuit

instance uniquely.

When two edges are combined, either through series or parallel reduction, the resulting edge is give a name that is the aggregate of the two edges names, combined with one of two symbols representing either a series (=) or parallel (|) reduction. These names are purely for debugging purposes, and are not used by the program at all. They do give some indication of how the S-P reduction has been carried out across the circuit. This can be seen in the example output in Code fragment 3. The current flow results for the circuit in Figure 5.7 can be easily obtained using the code in Code fragment 8 and have been formatted into table 5.8 where it can be seen that the flows have been generated as expected for this example circuit.

5.4.1 Comprehensive analysis of the test circuit

This section details a comprehensive analysis of the performance of the algorithms and implementation by using the circuit in Figure 5.5. The strategy is to simulate the circuit for all possible permutations of 5 resistance levels and analyse the results checking for the features we would expect to see for this circuit.

Initially the circuit in Figure 5.5 is created using the Circuit class API (with default zero resistance values) using the code in Code fragment 9.

Resistances `a` and `c` are arranged such that a `FORWARD` flow occurs from the `+` to `-` nodes. The parallel resistances `b`, `d`, `e` are connected such that a flow from `+` to `-` creates a `REVERSE` flow direction. This provides for an easy manual validation of the flow direction in the results since all the parallel resistances should have reverse flow. Once the flow direction is seen to be correct the flow direction provides an easy way of identifying the three parallel resistors when manually considering other aspects.

A simple test harness runs the new MCIRQ algorithm on every permutation of {`Zero`, `Lo`, `Med`, `Hi`, `Inf` } resistances for the above network of 5 resistances and results in 3125 simulations (5^5) which execute in less than a second on a modern laptop computer. This was achieved by creating the network once and using the `c.changeResistance(edge_name, resistancevalue);` to modify the resistance values. The circuit was then reduced for every permutation of resistances using `c.doSPReduction();` followed by a sequence of calls to

Code/Data fragment 8 Code for generating the main example circuit in Figure 5.7

```
Circuit c = new Circuit();

c.makeEdge("+", "a", Resistance.get("Zero"));
c.makeEdge("a", "b", Resistance.get("Zero"));
c.makeEdge("b", "c", Resistance.get("Zero"));
c.makeEdge("a", "d", Resistance.get("Lo"));
c.makeEdge("b", "e", Resistance.get("Lo"));
c.makeEdge("c", "f", Resistance.get("Lo"));
c.makeEdge("d", "g", Resistance.get("Lo"));
c.makeEdge("d", "h", Resistance.get("Lo"));
c.makeEdge("d", "i", Resistance.get("Lo"));
c.makeEdge("e", "j", Resistance.get("Med"));
c.makeEdge("e", "k", Resistance.get("Lo"));
c.makeEdge("f", "l", Resistance.get("Med"));
c.makeEdge("g", "m", Resistance.get("Med"));
c.makeEdge("h", "n", Resistance.get("Med"));
c.makeEdge("i", "n", Resistance.get("Lo"));
c.makeEdge("j", "o", Resistance.get("Med"));
c.makeEdge("j", "o", Resistance.get("Lo"));
c.makeEdge("k", "p", Resistance.get("Med"));
c.makeEdge("o", "p", Resistance.get("Zero"));
c.makeEdge("l", "q", Resistance.get("Lo"));
c.makeEdge("f", "q", Resistance.get("Lo"));
c.makeEdge("m", "r", Resistance.get("Hi"));
c.makeEdge("n", "r", Resistance.get("Med"));
c.makeEdge("r", "s", Resistance.get("Lo"));
c.makeEdge("p", "s", Resistance.get("Hi"));
c.makeEdge("q", "t", Resistance.get("Med"));
c.makeEdge("s", "u", Resistance.get("Lo"));
c.makeEdge("t", "v", Resistance.get("Inf"));
c.makeEdge("u", "w", Resistance.get("Hi"));
c.makeEdge("u", "w", Resistance.get("Med"));
c.makeEdge("v", "x", Resistance.get("Lo"));
c.makeEdge("w", "y", Resistance.get("Lo"));
c.makeEdge("x", "z", Resistance.get("Lo"));
c.makeEdge("y", "aa", Resistance.get("Zero"));
c.makeEdge("y", "ab", Resistance.get("Lo"));
c.makeEdge("y", "ac", Resistance.get("Med"));
c.makeEdge("z", "ad", Resistance.get("Lo"));
c.makeEdge("z", "ae", Resistance.get("Med"));
c.makeEdge("aa", "-", Resistance.get("Zero"));
c.makeEdge("ab", "aa", Resistance.get("Zero"));
c.makeEdge("ac", "ab", Resistance.get("Zero"));
c.makeEdge("ad", "ac", Resistance.get("Zero"));
c.makeEdge("ae", "ad", Resistance.get("Zero"));

c.doSPReduction(Resistance.get("Inf"));

ArrayList<Edge> edges = c.getTopEdges();
for (Edge e : edges){
    System.out.println(e.getName()+" "
        +e.getR()+" "
        +e.getV1().getName()+" "
        +e.getV2().getName()+" "
        +c.getCurrent(e));
}
```

Edge	Resistance	V1	V2	Flow
E0	Zero	+	a+b	I FORWARD
E1	Zero	a	b	i- FORWARD
E2	Zero	b	c	0 NONE
E3	Lo	a	d	I FORWARD
E4	Lo	b	e	i- FORWARD
E5	Lo	c	f	0 NONE
E6	Lo	d	g	i- FORWARD
E7	Lo	d	h	i- FORWARD
E8	Lo	d	i	I FORWARD
E9	Med	e	j	i- FORWARD
E10	Lo	e	k	i- FORWARD
E11	Med	f	l	0 NONE
E12	Med	g	m	i- FORWARD
E13	Med	h	n	i- FORWARD
E14	Lo	i	n	I FORWARD
E15	Med	j	o	i-- FORWARD
E16	Lo	j	o	i- FORWARD
E17	Med	k	p	i- FORWARD
E18	Zero	o	p	i- FORWARD
E19	Lo	l	q	0 NONE
E20	Lo	f	q	0 NONE
E21	Hi	m	r	i- FORWARD
E22	Med	n	r	I FORWARD
E23	Lo	r	s	I FORWARD
E24	Hi	p	s	i- FORWARD
E25	Med	q	t	0 NONE
E26	Lo	s	u	I FORWARD
E27	Inf	t	v	0 NONE
E28	Hi	u	w	i- FORWARD
E29	Med	u	w	I FORWARD
E30	Lo	v	x	0 NONE
E31	Lo	w	y	I FORWARD
E32	Lo	x	z	0 NONE
E33	Zero	y	aa	I FORWARD
E34	Lo	ab+y+aa	ab+y+aa	0 NONE
E35	Med	ab+y+aa	ac	0 NONE
E36	Lo	z	ad	0 NONE
E37	Med	z	ae	0 NONE
E38	Zero	ab+y+aa	-	I FORWARD
E39	Zero	ab	y+aa	0 NONE
E40	Zero	ac	ab+y+aa	0 NONE
E41	Zero	ad	ac	0 NONE
E42	Zero	ae	ad	0 NONE

Figure 5.8: Flow results for circuit in Figure 5.7

Code/Data fragment 9 Code for generating the circuit in Figure 5.5

```
c = new Circuit();
...
c.makeEdge("v2", "-",   "Zero", "c");
c.makeEdge("+", "v1", "Zero", "a");
c.makeEdge("v2", "v1", "Zero", "b");
c.makeEdge("v2", "v1", "Zero", "d");
c.makeEdge("v2", "v1", "Zero", "e");
```

`c.getCurrent(c.getTopEdge(edge_name))` method to obtain the flows for each edge.

While there are too many results to present explicitly for the entire set of simulation, some initial observations were made by importing the results to a spreadsheet and performing various sorting and analysis. Sorting the results by current flow from the source provides the information in table 5.9.

Entries	Flow	Direction	Note	
112	0	NONE	(FROM:v2 TO:v1)	Note: a and c are both `Inf`
661	I	FORWARD	(FROM + TO -)	Note single resistance is `Med`
931	i-	FORWARD	(FROM:+ TO:-)	Note single resistance is `Hi`
331	I+	FORWARD	(FROM:+ TO:-)	Note single resistance is `Lo`
61	M	FORWARD	(FROM:+ TO:-)	Note: `a` and `c` are both zero and also one of `b` or `d` or `e` is zero
1029	0		DISCONNECTED	Note: either: a is `Inf`, or `c` is `Inf`, or `b`,`d`,`e` are all `Inf`

Figure 5.9: Results of running the simulation for all permutations of resistance value for circuit in Figure 5.5 categorised by circuit flow

An extract of the results imported to the spreadsheet is shown in Figure 5.10. This table was sorted by the “flow, direction” column to categorise the results by flow from the source. The table shows the first and last three entries in each of the flow categories.

Within each category the resistance values are ordered such that all resistance values for resistance `e` are shown on adjacent rows for each permutation of `a`-`d`. This process is continued for each column moving left until for each category a set of rows exist for resistance a for each of its values.

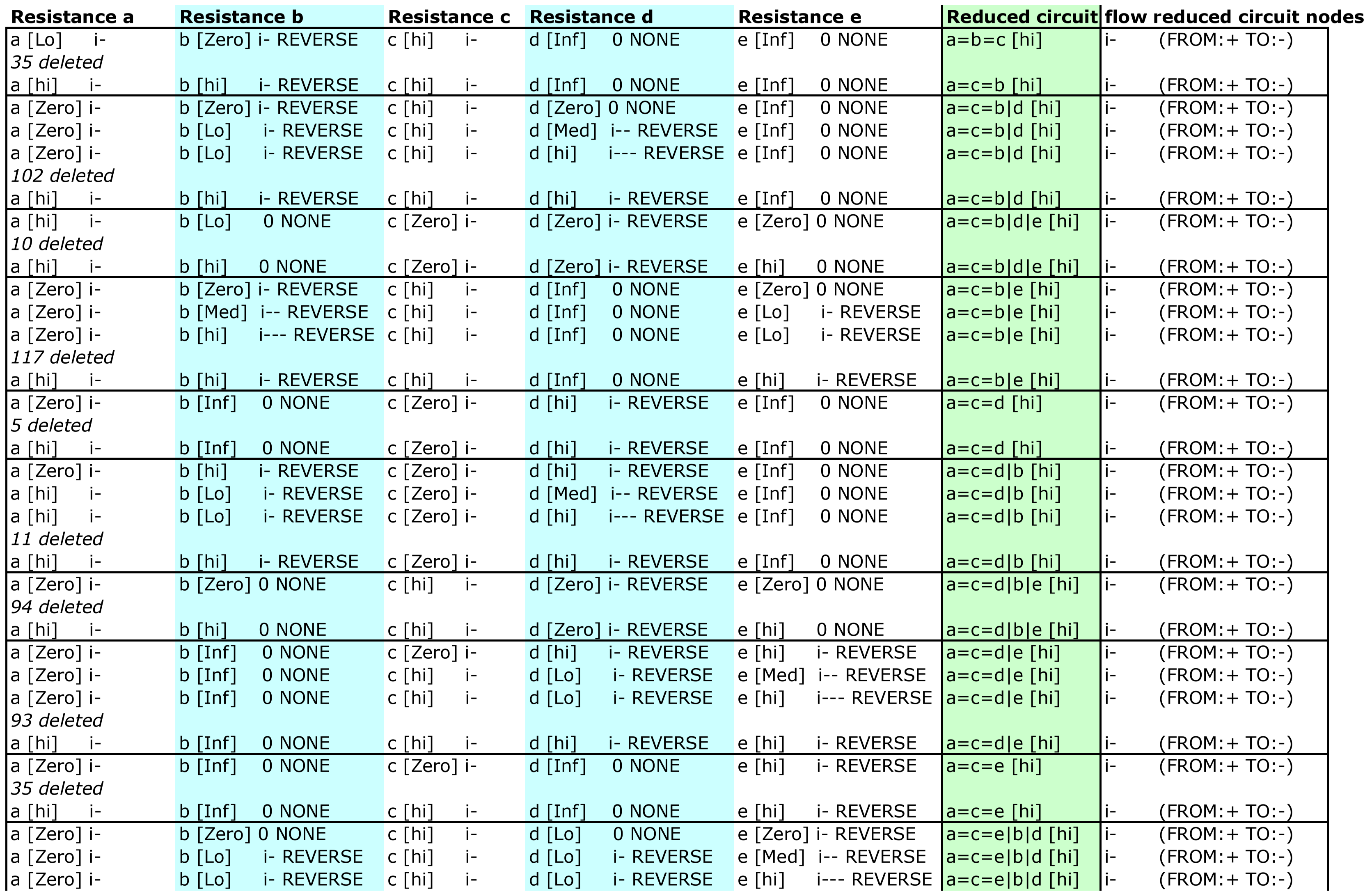
While it is infeasible to check each of the results manually, the observations in table 5.9 provide some confidence that the simulator is operating correctly. The first row illustrates that the circuit can collapse to a single resistance that does not connect to either of the terminal nodes only when `a` and `c` are both `Inf`. In this case the circuit collapses to a single resistor, but it is not connected to either source node and therefore can have no flow, has a floating voltage relative to either of the source nodes. When one of `a` or `c` is `Inf` we obtain a disconnected circuit, such that the collapsed circuit does not contain a single resistance between the terminal nodes. In this case the collapsed circuit is connected to one of the terminal nodes, unlike the previous case. The 5th case in table 5.9 provides maximal (`M`) level of current (a short circuit) and it is easily seen from the full set of results (and summarised in the table note) that this only occurs when `a` and `c` is `Zero` plus one of the parallel resistances is `Zero`. The remaining results provide three orders of flow magnitude and as expected these map directly to the three level of collapsed circuit resistance, as defined by the method. Figure 5.10 shows a truncated extract of the complete set of results from which the above observations were derived by simple observation. The figure shows the flow in each resistance `a` - `e` in the five leftmost columns, followed by the topology of the reduced circuit as described by the label assigned to the fully reduced circuit edge. The rightmost columns provide the source flow and direction together with the nodes that are connected by the reduced circuit.

Figures 5.11 and 5.12 expand the results from the part of table 5.10 for the case where the circuit flow level is `i-`. The aim of this figure is to show results for all of of the reduced circuit topologies generated by the analysis for `i-` since this is the system level flow that results from the greatest number of resistance permutations. The `i-` category was chosen because it provides the largest range of reduced circuit topologies. Several items were selected from each of the reduced circuit topologies present. Specifically the first and last simulations (according to the above resistance orderings) from each reduced circuit topology is shown together with the first row that shows an `i-`, `i--`and `i---` (where present).

By considering the reduced circuit column in this table together with the resistance values shown in square brackets in the first five columns in detail we can confirm the circuits are correctly collapsed and flows allocated. For example the first row illustrates the removal of resistance `d` and `e` (both `Inf`) resulting in a simple series circuit comprising `a`, `b` and `c`. The reduced circuit is actually `(a=(b=c))` however the brackets are omitted in the output since the resulting edge labels would become unwieldy and unhelpful for human use for larger circuits if brackets were included. The 4th and 5th non empty rows show the situation where part of the circuit has a qualitatively lower level of current flow than the flow between the source

Resistance a	Resistance b	Resistance c	Resistance d	Resistance e	Reduced circuit	flow, directio	reduced circuit nodes	
a [Inf] 0 NONE	b [Zero] 0 NONE	c [Inf] 0 NONE	d [Zero] 0 NONE	e [Zero] 0 NONE	b|d|e [Zero]	0 NONE	(FROM:v2 TO:v1)	
a [Inf] 0 NONE	b [Zero] 0 NONE	c [Inf] 0 NONE	d [Zero] 0 NONE	e [Lo] 0 NONE	b|d|e [Zero]	0 NONE	(FROM:v2 TO:v1)	
a [Inf] 0 NONE	b [Zero] 0 NONE	c [Inf] 0 NONE	d [Zero] 0 NONE	e [Med] 0 NONE	b|d|e [Zero]	0 NONE	(FROM:v2 TO:v1)	
								106 rows deleted
a [Inf] 0 NONE	b [Hi] 0 NONE	c [Inf] 0 NONE	d [Lo] 0 NONE	e [Zero] 0 NONE	e|d|b [Zero]	0 NONE	(FROM:v2 TO:v1)	
a [Inf] 0 NONE	b [Hi] 0 NONE	c [Inf] 0 NONE	d [Med] 0 NONE	e [Zero] 0 NONE	e|d|b [Zero]	0 NONE	(FROM:v2 TO:v1)	
a [Inf] 0 NONE	b [Hi] 0 NONE	c [Inf] 0 NONE	d [Hi] 0 NONE	e [Zero] 0 NONE	e|d|b [Zero]	0 NONE	(FROM:v2 TO:v1)	
a [Lo] I FORWARD	b [Zero] I REVERSE	c [Med] I FORWARD	d [Inf] 0 NONE	e [Inf] 0 NONE	a=b=c [Med]	I FORWARD	(FROM:+ TO:-)	
a [Med] I FORWARD	b [Zero] I REVERSE	c [Zero] I FORWARD	d [Inf] 0 NONE	e [Inf] 0 NONE	a=b=c [Med]	I FORWARD	(FROM:+ TO:-)	
a [Med] I FORWARD	b [Zero] I REVERSE	c [Lo] I FORWARD	d [Inf] 0 NONE	e [Inf] 0 NONE	a=b=c [Med]	I FORWARD	(FROM:+ TO:-)	
								657 rows deleted
a [Med] I FORWARD	b [Inf] 0 NONE	c [Med] I FORWARD	d [Zero] I REVERSE	e [Inf] 0 NONE	a=d=c [Med]	I FORWARD	(FROM:+ TO:-)	
a [Med] I FORWARD	b [Inf] 0 NONE	c [Med] I FORWARD	d [Lo] I REVERSE	e [Inf] 0 NONE	a=d=c [Med]	I FORWARD	(FROM:+ TO:-)	
a [Med] I FORWARD	b [Inf] 0 NONE	c [Med] I FORWARD	d [Med] I REVERSE	e [Inf] 0 NONE	a=d=c [Med]	I FORWARD	(FROM:+ TO:-)	
a [Lo] i- FORWARD	b [Zero] i- REVERSE	c [Hi] i- FORWARD	d [Inf] 0 NONE	e [Inf] 0 NONE	a=b=c [Hi]	i- FORWARD	(FROM:+ TO:-)	
a [Med] i- FORWARD	b [Zero] i- REVERSE	c [Hi] i- FORWARD	d [Inf] 0 NONE	e [Inf] 0 NONE	a=b=c [Hi]	i- FORWARD	(FROM:+ TO:-)	
a [Hi] i- FORWARD	b [Zero] i- REVERSE	c [Zero] i- FORWARD	d [Inf] 0 NONE	e [Inf] 0 NONE	a=b=c [Hi]	i- FORWARD	(FROM:+ TO:-)	
								925 rows deleted
a [Hi] i- FORWARD	b [Inf] 0 NONE	c [Hi] i- FORWARD	d [Lo] i- REVERSE	e [Inf] 0 NONE	a=d=c [Hi]	i- FORWARD	(FROM:+ TO:-)	
a [Hi] i- FORWARD	b [Inf] 0 NONE	c [Hi] i- FORWARD	d [Med] i- REVERSE	e [Inf] 0 NONE	a=d=c [Hi]	i- FORWARD	(FROM:+ TO:-)	
a [Hi] i- FORWARD	b [Inf] 0 NONE	c [Hi] i- FORWARD	d [Hi] i- REVERSE	e [Inf] 0 NONE	a=d=c [Hi]	i- FORWARD	(FROM:+ TO:-)	
a [Lo] I+ FORWARD	b [Zero] I+ REVERSE	c [Zero] I+ FORWARD	d [Inf] 0 NONE	e [Inf] 0 NONE	a=b=c [Lo]	I+ FORWARD	(FROM:+ TO:-)	
a [Lo] I+ FORWARD	b [Zero] I+ REVERSE	c [Lo] I+ FORWARD	d [Inf] 0 NONE	e [Inf] 0 NONE	a=b=c [Lo]	I+ FORWARD	(FROM:+ TO:-)	
a [Zero] I+ FORWARD	b [Zero] I+ REVERSE	c [Lo] I+ FORWARD	d [Inf] 0 NONE	e [Inf] 0 NONE	a=c=b [Lo]	I+ FORWARD	(FROM:+ TO:-)	
								325 rows deleted
a [Zero] I+ FORWARD	b [Inf] 0 NONE	c [Lo] I+ FORWARD	d [Lo] I+ REVERSE	e [Inf] 0 NONE	a=d=c [Lo]	I+ FORWARD	(FROM:+ TO:-)	
a [Lo] I+ FORWARD	b [Inf] 0 NONE	c [Lo] I+ FORWARD	d [Zero] I+ REVERSE	e [Inf] 0 NONE	a=d=c [Lo]	I+ FORWARD	(FROM:+ TO:-)	
a [Lo] I+ FORWARD	b [Inf] 0 NONE	c [Lo] I+ FORWARD	d [Lo] I+ REVERSE	e [Inf] 0 NONE	a=d=c [Lo]	I+ FORWARD	(FROM:+ TO:-)	
a [Zero] M FORWARD	b [Zero] M REVERSE	c [Zero] M FORWARD	d [Inf] 0 NONE	e [Inf] 0 NONE	a=c=b [Zero]	M FORWARD	(FROM:+ TO:-)	
a [Zero] M FORWARD	b [Zero] M REVERSE	c [Zero] M FORWARD	d [Zero] 0 NONE	e [Inf] 0 NONE	a=c=b|d [Zero]	M FORWARD	(FROM:+ TO:-)	
a [Zero] M FORWARD	b [Zero] M REVERSE	c [Zero] M FORWARD	d [Lo] 0 NONE	e [Inf] 0 NONE	a=c=b|d [Zero]	M FORWARD	(FROM:+ TO:-)	
								55 rows deleted
a [Zero] M FORWARD	b [Hi] 0 NONE	c [Zero] M FORWARD	d [Lo] 0 NONE	e [Zero] M REVERSE	a=c=e|d|b [Zero]	M FORWARD	(FROM:+ TO:-)	
a [Zero] M FORWARD	b [Hi] 0 NONE	c [Zero] M FORWARD	d [Med] 0 NONE	e [Zero] M REVERSE	a=c=e|d|b [Zero]	M FORWARD	(FROM:+ TO:-)	
a [Zero] M FORWARD	b [Hi] 0 NONE	c [Zero] M FORWARD	d [Hi] 0 NONE	e [Zero] M REVERSE	a=c=e|d|b [Zero]	M FORWARD	(FROM:+ TO:-)	
a [Zero] 0 NONE	b [Zero] 0 NONE	c [Inf] 0 NONE	d [Zero] 0 NONE	e [Zero] 0 NONE	NO SINGLE RESISTANCE - DISCONNECTED			
a [Zero] 0 NONE	b [Zero] 0 NONE	c [Inf] 0 NONE	d [Zero] 0 NONE	e [Lo] 0 NONE	NO SINGLE RESISTANCE - DISCONNECTED			
a [Zero] 0 NONE	b [Zero] 0 NONE	c [Inf] 0 NONE	d [Zero] 0 NONE	e [Med] 0 NONE	NO SINGLE RESISTANCE - DISCONNECTED			
								1023 rows deleted
a [Inf] 0 NONE	b [Inf] 0 NONE	c [Inf] 0 NONE	d [Inf] 0 NONE	e [Med] 0 NONE	NO SINGLE RESISTANCE - DISCONNECTED			
a [Inf] 0 NONE	b [Inf] 0 NONE	c [Inf] 0 NONE	d [Inf] 0 NONE	e [Hi] 0 NONE	NO SINGLE RESISTANCE - DISCONNECTED			
a [Inf] 0 NONE	b [Inf] 0 NONE	c [Inf] 0 NONE	d [Inf] 0 NONE	e [Inf] 0 NONE	NO SINGLE RESISTANCE - DISCONNECTED			

Figure 5.10: simplecircuitoutputselected



Resistance a	Resistance b	Resistance c	Resistance d	Resistance e	Reduced circuit	flow reduced circuit nodes
a [Lo] i-	b [Zero] i- REVERSE	c [hi] i-	d [Inf] 0 NONE	e [Inf] 0 NONE	a=b=c [hi]	i- (FROM:+ TO:-)
35 deleted						
a [hi] i-	b [hi] i- REVERSE	c [hi] i-	d [Inf] 0 NONE	e [Inf] 0 NONE	a=c=b [hi]	i- (FROM:+ TO:-)
a [Zero] i-	b [Zero] i- REVERSE	c [hi] i-	d [Zero] 0 NONE	e [Inf] 0 NONE	a=c=b|d [hi]	i- (FROM:+ TO:-)
a [Zero] i-	b [Lo] i- REVERSE	c [hi] i-	d [Med] i-- REVERSE	e [Inf] 0 NONE	a=c=b|d [hi]	i- (FROM:+ TO:-)
a [Zero] i-	b [Lo] i- REVERSE	c [hi] i-	d [hi] i--- REVERSE	e [Inf] 0 NONE	a=c=b|d [hi]	i- (FROM:+ TO:-)
102 deleted						
a [hi] i-	b [hi] i- REVERSE	c [hi] i-	d [hi] i- REVERSE	e [Inf] 0 NONE	a=c=b|d [hi]	i- (FROM:+ TO:-)
a [hi] i-	b [Lo] 0 NONE	c [Zero] i-	d [Zero] i- REVERSE	e [Zero] 0 NONE	a=c=b|d|e [hi]	i- (FROM:+ TO:-)
10 deleted						
a [hi] i-	b [hi] 0 NONE	c [Zero] i-	d [Zero] i- REVERSE	e [hi] 0 NONE	a=c=b|d|e [hi]	i- (FROM:+ TO:-)
a [Zero] i-	b [Zero] i- REVERSE	c [hi] i-	d [Inf] 0 NONE	e [Zero] 0 NONE	a=c=b|e [hi]	i- (FROM:+ TO:-)
a [Zero] i-	b [Med] i-- REVERSE	c [hi] i-	d [Inf] 0 NONE	e [Lo] i- REVERSE	a=c=b|e [hi]	i- (FROM:+ TO:-)
a [Zero] i-	b [hi] i--- REVERSE	c [hi] i-	d [Inf] 0 NONE	e [Lo] i- REVERSE	a=c=b|e [hi]	i- (FROM:+ TO:-)
117 deleted						
a [hi] i-	b [hi] i- REVERSE	c [hi] i-	d [Inf] 0 NONE	e [hi] i- REVERSE	a=c=b|e [hi]	i- (FROM:+ TO:-)
a [Zero] i-	b [Inf] 0 NONE	c [Zero] i-	d [hi] i- REVERSE	e [Inf] 0 NONE	a=c=d [hi]	i- (FROM:+ TO:-)
5 deleted						
a [hi] i-	b [Inf] 0 NONE	c [Zero] i-	d [hi] i- REVERSE	e [Inf] 0 NONE	a=c=d [hi]	i- (FROM:+ TO:-)
a [Zero] i-	b [hi] i- REVERSE	c [Zero] i-	d [hi] i- REVERSE	e [Inf] 0 NONE	a=c=d|b [hi]	i- (FROM:+ TO:-)
a [hi] i-	b [Lo] i- REVERSE	c [Zero] i-	d [Med] i-- REVERSE	e [Inf] 0 NONE	a=c=d|b [hi]	i- (FROM:+ TO:-)
a [hi] i-	b [Lo] i- REVERSE	c [Zero] i-	d [hi] i--- REVERSE	e [Inf] 0 NONE	a=c=d|b [hi]	i- (FROM:+ TO:-)
11 deleted						
a [hi] i-	b [hi] i- REVERSE	c [Zero] i-	d [hi] i- REVERSE	e [Inf] 0 NONE	a=c=d|b [hi]	i- (FROM:+ TO:-)
a [Zero] i-	b [Zero] 0 NONE	c [hi] i-	d [Zero] i- REVERSE	e [Zero] 0 NONE	a=c=d|b|e [hi]	i- (FROM:+ TO:-)
94 deleted						
a [hi] i-	b [hi] 0 NONE	c [hi] i-	d [Zero] i- REVERSE	e [hi] 0 NONE	a=c=d|b|e [hi]	i- (FROM:+ TO:-)
a [Zero] i-	b [Inf] 0 NONE	c [Zero] i-	d [hi] i- REVERSE	e [hi] i- REVERSE	a=c=d|e [hi]	i- (FROM:+ TO:-)
a [Zero] i-	b [Inf] 0 NONE	c [hi] i-	d [Lo] i- REVERSE	e [Med] i-- REVERSE	a=c=d|e [hi]	i- (FROM:+ TO:-)
a [Zero] i-	b [Inf] 0 NONE	c [hi] i-	d [Lo] i- REVERSE	e [hi] i--- REVERSE	a=c=d|e [hi]	i- (FROM:+ TO:-)
93 deleted						
a [hi] i-	b [Inf] 0 NONE	c [hi] i-	d [hi] i- REVERSE	e [hi] i- REVERSE	a=c=d|e [hi]	i- (FROM:+ TO:-)
a [Zero] i-	b [Inf] 0 NONE	c [Zero] i-	d [Inf] 0 NONE	e [hi] i- REVERSE	a=c=e [hi]	i- (FROM:+ TO:-)
35 deleted						
a [hi] i-	b [Inf] 0 NONE	c [hi] i-	d [Inf] 0 NONE	e [hi] i- REVERSE	a=c=e [hi]	i- (FROM:+ TO:-)
a [Zero] i-	b [Zero] 0 NONE	c [hi] i-	d [Lo] 0 NONE	e [Zero] i- REVERSE	a=c=e|b|d [hi]	i- (FROM:+ TO:-)
a [Zero] i-	b [Lo] i- REVERSE	c [hi] i-	d [Lo] i- REVERSE	e [Med] i-- REVERSE	a=c=e|b|d [hi]	i- (FROM:+ TO:-)
a [Zero] i-	b [Lo] i- REVERSE	c [hi] i-	d [Lo] i- REVERSE	e [hi] i--- REVERSE	a=c=e|b|d [hi]	i- (FROM:+ TO:-)

Figure 5.11: simplecircuitoutputselectedtopology1

306 deleted												
a [hi]	i-	b [hi]	i- REVERSE	c [hi]	i-	d [hi]	i- REVERSE	e [hi]	i- REVERSE	a=c=e|b|d [hi]	i-	(FROM:+ TO:-)
a [Zero]	i-	b [Inf]	0 NONE	c [hi]	i-	d [Zero]	0 NONE	e [Zero]	i- REVERSE	a=c=e|d [hi]	i-	(FROM:+ TO:-)
22 deleted												
a [hi]	i-	b [Inf]	0 NONE	c [hi]	i-	d [Zero]	i- REVERSE	e [hi]	0 NONE	a=c=e|d [hi]	i-	(FROM:+ TO:-)
a [Zero]	i-	b [hi]	i- REVERSE	c [Zero]	i-	d [hi]	i- REVERSE	e [hi]	i- REVERSE	a=c=e|d|b [hi]	i-	(FROM:+ TO:-)
a [hi]	i-	b [Lo]	0 NONE	c [Zero]	i-	d [Lo]	0 NONE	e [Zero]	i- REVERSE	a=c=e|d|b [hi]	i-	(FROM:+ TO:-)
a [hi]	i-	b [Lo]	i- REVERSE	c [Zero]	i-	d [Lo]	i- REVERSE	e [Med]	i-- REVERSE	a=c=e|d|b [hi]	i-	(FROM:+ TO:-)
a [hi]	i-	b [Lo]	i- REVERSE	c [Zero]	i-	d [Lo]	i- REVERSE	e [hi]	i--- REVERSE	a=c=e|d|b [hi]	i-	(FROM:+ TO:-)
34 deleted												
a [hi]	i-	b [hi]	i- REVERSE	c [Zero]	i-	d [hi]	i- REVERSE	e [hi]	i- REVERSE	a=c=e|d|b [hi]	i-	(FROM:+ TO:-)
a [Zero]	i-	b [Inf]	0 NONE	c [Lo]	i-	d [hi]	i- REVERSE	e [Inf]	0 NONE	a=d=c [hi]	i-	(FROM:+ TO:-)
28 deleted												
a [hi]	i-	b [Inf]	0 NONE	c [hi]	i-	d [hi]	i- REVERSE	e [Inf]	0 NONE	a=d=c [hi]	i-	(FROM:+ TO:-)

Figure 5.12: simplecircuitoutputselectedtopology2

terminals, demonstrating that flow distribution between parallel segments works correctly. In the 4th row `b` is `Lo` with a `i-` flow and is in parallel with `d` which is `Med`, thus causing an order of magnitude lower flow in `d` then `b` of `i--`. In the 5th row `b` is again `Lo` with a `i-` flow and is in parallel with `d` which this time is `Hi`, causing a two order of magnitude lower flow of `i---`.

The previous tests show that zero and infinite resistances are handled correctly resulting in changes to the circuit topology due to disconnected and shorted circuit fragments. To validate the multiple qualitative resistance topologies the circuit was simulated again with the following constraints:

- Resistance `c` is fixed at `Zero`. Since `c` and `a` are in series the greater resistance will act as a current limiter providing a symmetrical result for either.
- Resistances `a`, `b`, `d`, `e` are considered with all permutations of the possible values {`Lo`, `Med`, `Hi`}

This simulation requires all permutations 3 resistance levels for 4 resistances and hence generates the $3^4 = 81$ results all of which are presented in Figures 5.13 and 5.14. The lack of `Zero` and `Inf` resistances in this test mean that all the circuits collapse in the same way to the configuration `a=c=e|b|d`. This table can be analysed in detail to manually check the result of a wide variety of circuit configurations. Specifically the lower levels of flow that are automatically introduced by the simulator because of parallel flow distribution can be observed in the individual parallel circuit resistances (`b`, `d`, `e`). For example the second row of Figure 5.13 illustrates a high resistance (`e`) in parallel with medium ones (`b` and `d`) carrying a flow of `I`, thus `e` has a flow of `i-`, one order of magnitude less than `I` .

Considering the table as a whole we observe (as expected) that an overall circuit flow of `I` provides a minimum flow in any of the parallel branched of `i--`, since the maximum difference in resistance between parallel branched is two. For an overall circuit flow of `i-` the minimum is again two orders lower, for the same reason, and this is borne out by the smallest `i---` values in the second section of the table. The final section of the table in Figure 5.14 shows that for an overall circuit flow of `I+` the minimum branch flow is `i-` (again two orders lower), and also that resistances `a` and `c` must have a maximum resistance of `Lo` (`c` is fixed at `Zero` anyway) to allow a flow of `I+`.

Resistance a	Resistance b	Resistance c	Resistance d	Resistance e	Reduced circuit	flow	reduced circuit node
a [Lo] I	b [Med] I REVERSE	c [Zero] I	d [Med] I REVERSE	e [Med] I REVERSE	a=c=e|b|d [Med]	I	(FROM:+ TO:-)
a [Lo] I	b [Med] I REVERSE	c [Zero] I	d [Med] I REVERSE	e [Hi] i- REVERSE	a=c=e|b|d [Med]	I	(FROM:+ TO:-)
a [Lo] I	b [Med] I REVERSE	c [Zero] I	d [Hi] i- REVERSE	e [Med] I REVERSE	a=c=e|b|d [Med]	I	(FROM:+ TO:-)
a [Lo] I	b [Med] I REVERSE	c [Zero] I	d [Hi] i- REVERSE	e [Hi] i- REVERSE	a=c=e|b|d [Med]	I	(FROM:+ TO:-)
a [Lo] I	b [Hi] i- REVERSE	c [Zero] I	d [Med] I REVERSE	e [Med] I REVERSE	a=c=e|b|d [Med]	I	(FROM:+ TO:-)
a [Lo] I	b [Hi] i- REVERSE	c [Zero] I	d [Med] I REVERSE	e [Hi] i- REVERSE	a=c=e|b|d [Med]	I	(FROM:+ TO:-)
a [Lo] I	b [Hi] i- REVERSE	c [Zero] I	d [Hi] i- REVERSE	e [Med] I REVERSE	a=c=e|b|d [Med]	I	(FROM:+ TO:-)
a [Med] I	b [Lo] I REVERSE	c [Zero] I	d [Lo] I REVERSE	e [Lo] I REVERSE	a=c=e|b|d [Med]	I	(FROM:+ TO:-)
a [Med] I	b [Lo] I REVERSE	c [Zero] I	d [Lo] I REVERSE	e [Med] i- REVERSE	a=c=e|b|d [Med]	I	(FROM:+ TO:-)
a [Med] I	b [Lo] I REVERSE	c [Zero] I	d [Lo] I REVERSE	e [Hi] i-- REVERSE	a=c=e|b|d [Med]	I	(FROM:+ TO:-)
a [Med] I	b [Lo] I REVERSE	c [Zero] I	d [Med] i- REVERSE	e [Lo] I REVERSE	a=c=e|b|d [Med]	I	(FROM:+ TO:-)
a [Med] I	b [Lo] I REVERSE	c [Zero] I	d [Med] i- REVERSE	e [Med] i- REVERSE	a=c=e|b|d [Med]	I	(FROM:+ TO:-)
a [Med] I	b [Lo] I REVERSE	c [Zero] I	d [Med] i- REVERSE	e [Hi] i-- REVERSE	a=c=e|b|d [Med]	I	(FROM:+ TO:-)
a [Med] I	b [Lo] I REVERSE	c [Zero] I	d [Hi] i-- REVERSE	e [Lo] I REVERSE	a=c=e|b|d [Med]	I	(FROM:+ TO:-)
a [Med] I	b [Lo] I REVERSE	c [Zero] I	d [Hi] i-- REVERSE	e [Med] i- REVERSE	a=c=e|b|d [Med]	I	(FROM:+ TO:-)
a [Med] I	b [Lo] I REVERSE	c [Zero] I	d [Hi] i-- REVERSE	e [Hi] i-- REVERSE	a=c=e|b|d [Med]	I	(FROM:+ TO:-)
a [Med] I	b [Med] i- REVERSE	c [Zero] I	d [Lo] I REVERSE	e [Lo] I REVERSE	a=c=e|b|d [Med]	I	(FROM:+ TO:-)
a [Med] I	b [Med] i- REVERSE	c [Zero] I	d [Lo] I REVERSE	e [Med] i- REVERSE	a=c=e|b|d [Med]	I	(FROM:+ TO:-)
a [Med] I	b [Med] i- REVERSE	c [Zero] I	d [Lo] I REVERSE	e [Hi] i-- REVERSE	a=c=e|b|d [Med]	I	(FROM:+ TO:-)
a [Med] I	b [Med] i- REVERSE	c [Zero] I	d [Med] i- REVERSE	e [Lo] I REVERSE	a=c=e|b|d [Med]	I	(FROM:+ TO:-)
a [Med] I	b [Med] I REVERSE	c [Zero] I	d [Med] I REVERSE	e [Med] I REVERSE	a=c=e|b|d [Med]	I	(FROM:+ TO:-)
a [Med] I	b [Med] I REVERSE	c [Zero] I	d [Med] I REVERSE	e [Hi] i- REVERSE	a=c=e|b|d [Med]	I	(FROM:+ TO:-)
a [Med] I	b [Med] i- REVERSE	c [Zero] I	d [Hi] i-- REVERSE	e [Lo] I REVERSE	a=c=e|b|d [Med]	I	(FROM:+ TO:-)
a [Med] I	b [Med] I REVERSE	c [Zero] I	d [Hi] i- REVERSE	e [Med] I REVERSE	a=c=e|b|d [Med]	I	(FROM:+ TO:-)
a [Med] I	b [Med] I REVERSE	c [Zero] I	d [Hi] i- REVERSE	e [Hi] i- REVERSE	a=c=e|b|d [Med]	I	(FROM:+ TO:-)
a [Med] I	b [Hi] i-- REVERSE	c [Zero] I	d [Lo] I REVERSE	e [Lo] I REVERSE	a=c=e|b|d [Med]	I	(FROM:+ TO:-)
a [Med] I	b [Hi] i-- REVERSE	c [Zero] I	d [Lo] I REVERSE	e [Med] i- REVERSE	a=c=e|b|d [Med]	I	(FROM:+ TO:-)
a [Med] I	b [Hi] i-- REVERSE	c [Zero] I	d [Lo] I REVERSE	e [Hi] i-- REVERSE	a=c=e|b|d [Med]	I	(FROM:+ TO:-)
a [Med] I	b [Hi] i-- REVERSE	c [Zero] I	d [Med] i- REVERSE	e [Lo] I REVERSE	a=c=e|b|d [Med]	I	(FROM:+ TO:-)
a [Med] I	b [Hi] i- REVERSE	c [Zero] I	d [Med] I REVERSE	e [Med] I REVERSE	a=c=e|b|d [Med]	I	(FROM:+ TO:-)
a [Med] I	b [Hi] i- REVERSE	c [Zero] I	d [Med] I REVERSE	e [Hi] i- REVERSE	a=c=e|b|d [Med]	I	(FROM:+ TO:-)
a [Med] I	b [Hi] i-- REVERSE	c [Zero] I	d [Hi] i-- REVERSE	e [Lo] I REVERSE	a=c=e|b|d [Med]	I	(FROM:+ TO:-)
a [Med] I	b [Hi] i- REVERSE	c [Zero] I	d [Hi] i- REVERSE	e [Med] I REVERSE	a=c=e|b|d [Med]	I	(FROM:+ TO:-)
a [Lo] i-	b [Hi] i- REVERSE	c [Zero] i-	d [Hi] i- REVERSE	e [Hi] i- REVERSE	a=c=e|b|d [Hi]	i-	(FROM:+ TO:-)
a [Med] i-	b [Hi] i- REVERSE	c [Zero] i-	d [Hi] i- REVERSE	e [Hi] i- REVERSE	a=c=e|b|d [Hi]	i-	(FROM:+ TO:-)
a [Hi] i-	b [Lo] i- REVERSE	c [Zero] i-	d [Lo] i- REVERSE	e [Lo] i- REVERSE	a=c=e|b|d [Hi]	i-	(FROM:+ TO:-)
a [Hi] i-	b [Lo] i- REVERSE	c [Zero] i-	d [Lo] i- REVERSE	e [Med] i-- REVERSE	a=c=e|b|d [Hi]	i-	(FROM:+ TO:-)
a [Hi] i-	b [Lo] i- REVERSE	c [Zero] i-	d [Lo] i- REVERSE	e [Hi] i--- REVERSE	a=c=e|b|d [Hi]	i-	(FROM:+ TO:-)
a [Hi] i-	b [Lo] i- REVERSE	c [Zero] i-	d [Med] i-- REVERSE	e [Lo] i- REVERSE	a=c=e|b|d [Hi]	i-	(FROM:+ TO:-)
a [Hi] i-	b [Lo] i- REVERSE	c [Zero] i-	d [Med] i-- REVERSE	e [Med] i-- REVERSE	a=c=e|b|d [Hi]	i-	(FROM:+ TO:-)
a [Hi] i-	b [Lo] i- REVERSE	c [Zero] i-	d [Med] i-- REVERSE	e [Hi] i--- REVERSE	a=c=e|b|d [Hi]	i-	(FROM:+ TO:-)

Figure 5.13: simplesimplecircuitlomedhi1

a [Hi]	i-	b [Lo]	i- REVERSE	c [Zero]	i-	d [Hi]	i--- REVERSE	e [Lo]	i- REVERSE	a=c=e|b|d [Hi]	i-	(FROM:+ TO:-)
a [Hi]	i-	b [Lo]	i- REVERSE	c [Zero]	i-	d [Hi]	i--- REVERSE	e [Med]	i-- REVERSE	a=c=e|b|d [Hi]	i-	(FROM:+ TO:-)
a [Hi]	i-	b [Lo]	i- REVERSE	c [Zero]	i-	d [Hi]	i--- REVERSE	e [Hi]	i--- REVERSE	a=c=e|b|d [Hi]	i-	(FROM:+ TO:-)
a [Hi]	i-	b [Med]	i-- REVERSE	c [Zero]	i-	d [Lo]	i- REVERSE	e [Lo]	i- REVERSE	a=c=e|b|d [Hi]	i-	(FROM:+ TO:-)
a [Hi]	i-	b [Med]	i-- REVERSE	c [Zero]	i-	d [Lo]	i- REVERSE	e [Med]	i-- REVERSE	a=c=e|b|d [Hi]	i-	(FROM:+ TO:-)
a [Hi]	i-	b [Med]	i-- REVERSE	c [Zero]	i-	d [Lo]	i- REVERSE	e [Hi]	i--- REVERSE	a=c=e|b|d [Hi]	i-	(FROM:+ TO:-)
a [Hi]	i-	b [Med]	i-- REVERSE	c [Zero]	i-	d [Med]	i-- REVERSE	e [Lo]	i- REVERSE	a=c=e|b|d [Hi]	i-	(FROM:+ TO:-)
a [Hi]	i-	b [Med]	i- REVERSE	c [Zero]	i-	d [Med]	i- REVERSE	e [Med]	i- REVERSE	a=c=e|b|d [Hi]	i-	(FROM:+ TO:-)
a [Hi]	i-	b [Med]	i- REVERSE	c [Zero]	i-	d [Med]	i- REVERSE	e [Hi]	i-- REVERSE	a=c=e|b|d [Hi]	i-	(FROM:+ TO:-)
a [Hi]	i-	b [Med]	i-- REVERSE	c [Zero]	i-	d [Hi]	i--- REVERSE	e [Lo]	i- REVERSE	a=c=e|b|d [Hi]	i-	(FROM:+ TO:-)
a [Hi]	i-	b [Med]	i- REVERSE	c [Zero]	i-	d [Hi]	i-- REVERSE	e [Med]	i- REVERSE	a=c=e|b|d [Hi]	i-	(FROM:+ TO:-)
a [Hi]	i-	b [Med]	i- REVERSE	c [Zero]	i-	d [Hi]	i-- REVERSE	e [Hi]	i-- REVERSE	a=c=e|b|d [Hi]	i-	(FROM:+ TO:-)
a [Hi]	i-	b [Hi]	i--- REVERSE	c [Zero]	i-	d [Lo]	i- REVERSE	e [Lo]	i- REVERSE	a=c=e|b|d [Hi]	i-	(FROM:+ TO:-)
a [Hi]	i-	b [Hi]	i--- REVERSE	c [Zero]	i-	d [Lo]	i- REVERSE	e [Med]	i-- REVERSE	a=c=e|b|d [Hi]	i-	(FROM:+ TO:-)
a [Hi]	i-	b [Hi]	i--- REVERSE	c [Zero]	i-	d [Lo]	i- REVERSE	e [Hi]	i--- REVERSE	a=c=e|b|d [Hi]	i-	(FROM:+ TO:-)
a [Hi]	i-	b [Hi]	i--- REVERSE	c [Zero]	i-	d [Med]	i-- REVERSE	e [Lo]	i- REVERSE	a=c=e|b|d [Hi]	i-	(FROM:+ TO:-)
a [Hi]	i-	b [Hi]	i-- REVERSE	c [Zero]	i-	d [Med]	i- REVERSE	e [Med]	i- REVERSE	a=c=e|b|d [Hi]	i-	(FROM:+ TO:-)
a [Hi]	i-	b [Hi]	i-- REVERSE	c [Zero]	i-	d [Med]	i- REVERSE	e [Hi]	i-- REVERSE	a=c=e|b|d [Hi]	i-	(FROM:+ TO:-)
a [Hi]	i-	b [Hi]	i--- REVERSE	c [Zero]	i-	d [Hi]	i--- REVERSE	e [Lo]	i- REVERSE	a=c=e|b|d [Hi]	i-	(FROM:+ TO:-)
a [Hi]	i-	b [Hi]	i-- REVERSE	c [Zero]	i-	d [Hi]	i-- REVERSE	e [Med]	i- REVERSE	a=c=e|b|d [Hi]	i-	(FROM:+ TO:-)
a [Hi]	i-	b [Hi]	i- REVERSE	c [Zero]	i-	d [Hi]	i- REVERSE	e [Hi]	i- REVERSE	a=c=e|b|d [Hi]	i-	(FROM:+ TO:-)
a [Lo]	I+	b [Lo]	I+ REVERSE	c [Zero]	I+	d [Lo]	I+ REVERSE	e [Lo]	I+ REVERSE	a=c=e|b|d [Lo]	I+	(FROM:+ TO:-)
a [Lo]	I+	b [Lo]	I+ REVERSE	c [Zero]	I+	d [Lo]	I+ REVERSE	e [Med]	I REVERSE	a=c=e|b|d [Lo]	I+	(FROM:+ TO:-)
a [Lo]	I+	b [Lo]	I+ REVERSE	c [Zero]	I+	d [Lo]	I+ REVERSE	e [Hi]	i- REVERSE	a=c=e|b|d [Lo]	I+	(FROM:+ TO:-)
a [Lo]	I+	b [Lo]	I+ REVERSE	c [Zero]	I+	d [Med]	I REVERSE	e [Lo]	I+ REVERSE	a=c=e|b|d [Lo]	I+	(FROM:+ TO:-)
a [Lo]	I+	b [Lo]	I+ REVERSE	c [Zero]	I+	d [Med]	I REVERSE	e [Med]	I REVERSE	a=c=e|b|d [Lo]	I+	(FROM:+ TO:-)
a [Lo]	I+	b [Lo]	I+ REVERSE	c [Zero]	I+	d [Med]	I REVERSE	e [Hi]	i- REVERSE	a=c=e|b|d [Lo]	I+	(FROM:+ TO:-)
a [Lo]	I+	b [Lo]	I+ REVERSE	c [Zero]	I+	d [Hi]	i- REVERSE	e [Lo]	I+ REVERSE	a=c=e|b|d [Lo]	I+	(FROM:+ TO:-)
a [Lo]	I+	b [Lo]	I+ REVERSE	c [Zero]	I+	d [Hi]	i- REVERSE	e [Med]	I REVERSE	a=c=e|b|d [Lo]	I+	(FROM:+ TO:-)
a [Lo]	I+	b [Lo]	I+ REVERSE	c [Zero]	I+	d [Hi]	i- REVERSE	e [Hi]	i- REVERSE	a=c=e|b|d [Lo]	I+	(FROM:+ TO:-)
a [Lo]	I+	b [Med]	I REVERSE	c [Zero]	I+	d [Lo]	I+ REVERSE	e [Lo]	I+ REVERSE	a=c=e|b|d [Lo]	I+	(FROM:+ TO:-)
a [Lo]	I+	b [Med]	I REVERSE	c [Zero]	I+	d [Lo]	I+ REVERSE	e [Med]	I REVERSE	a=c=e|b|d [Lo]	I+	(FROM:+ TO:-)
a [Lo]	I+	b [Med]	I REVERSE	c [Zero]	I+	d [Lo]	I+ REVERSE	e [Hi]	i- REVERSE	a=c=e|b|d [Lo]	I+	(FROM:+ TO:-)
a [Lo]	I+	b [Med]	I REVERSE	c [Zero]	I+	d [Med]	I REVERSE	e [Lo]	I+ REVERSE	a=c=e|b|d [Lo]	I+	(FROM:+ TO:-)
a [Lo]	I+	b [Med]	I REVERSE	c [Zero]	I+	d [Hi]	i- REVERSE	e [Lo]	I+ REVERSE	a=c=e|b|d [Lo]	I+	(FROM:+ TO:-)
a [Lo]	I+	b [Hi]	i- REVERSE	c [Zero]	I+	d [Lo]	I+ REVERSE	e [Lo]	I+ REVERSE	a=c=e|b|d [Lo]	I+	(FROM:+ TO:-)
a [Lo]	I+	b [Hi]	i- REVERSE	c [Zero]	I+	d [Lo]	I+ REVERSE	e [Med]	I REVERSE	a=c=e|b|d [Lo]	I+	(FROM:+ TO:-)
a [Lo]	I+	b [Hi]	i- REVERSE	c [Zero]	I+	d [Lo]	I+ REVERSE	e [Hi]	i- REVERSE	a=c=e|b|d [Lo]	I+	(FROM:+ TO:-)
a [Lo]	I+	b [Hi]	i- REVERSE	c [Zero]	I+	d [Med]	I REVERSE	e [Lo]	I+ REVERSE	a=c=e|b|d [Lo]	I+	(FROM:+ TO:-)
a [Lo]	I+	b [Hi]	i- REVERSE	c [Zero]	I+	d [Hi]	i- REVERSE	e [Lo]	I+ REVERSE	a=c=e|b|d [Lo]	I+	(FROM:+ TO:-)

Figure 5.14: simplesimplecircuitlomedhi2

The analysis of the comprehensive set of simulations for circuit 5.5(b) combined with the use of the tool in higher level tasks such as FMEA for complex circuits such as the one in Figure 2.3 described in the next section provide a high level of confidence in the tool and the algorithms it implements.

5.5 Deployment of the New Simulator

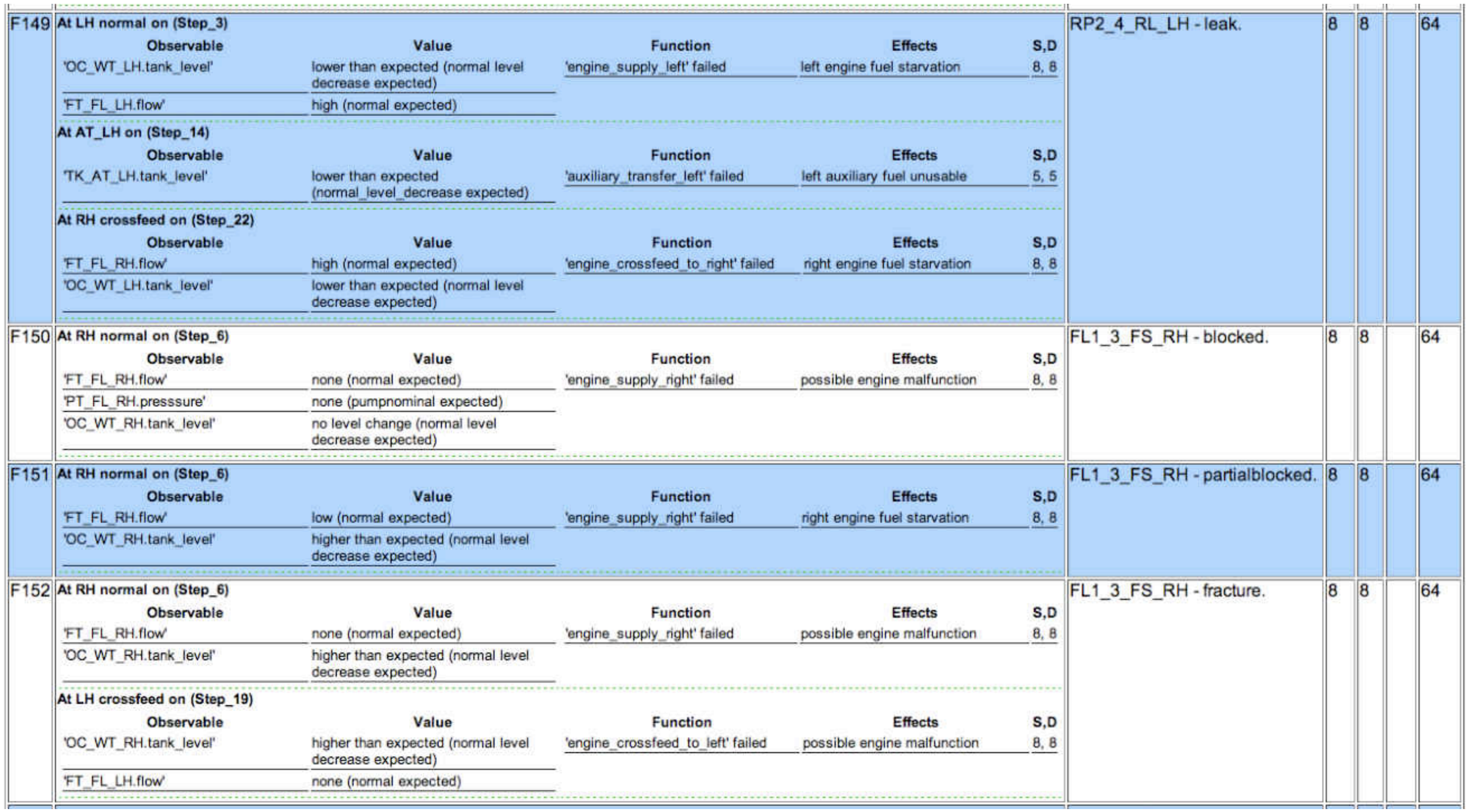
The simulator developed in this thesis has been successfully deployed not only for electrical systems presented in this work, but also for fluid flow systems. One specific project (Autonomous Systems Technology Related Airborne Evaluation & Assessment [$ASTRAEA$]) addressed aircraft fuel systems and has developed several tools that use the simulator. These tools include an automated FMEA tool, a symptom generation tool, an on board diagnosis system and a diagnosability assessment and sensor placement tool. These tools all require extensive use of the simulator as they consider a wide range of nominal operating modes and failure modes. The failure modes in particular often produce circuits with very abnormal topologies and are hence a good test of the ability of the simulator to produce the correct behaviour for extreme topologies. An example FMEA analysis fragment produced by the ASTRAEA project is shown in Figure 5.15 for the system in Figure 2.3.

When modelling sophisticated fluid flow systems using the new MCIRQ simulator it was found that some additional capability is needed that was not originally provided in the work undertaken in this thesis, however it has been pleasing to find that integration of these capabilities by extension of the implementation was possible. For general interest and as a demonstration of the utility and extensibility of the simulator a brief description of two major implications of the use of the simulator for the aircraft fuel systems are itemised below:

- Multiple sources (pumps) can exist in the same network. This problem was solved by running the simulator multiple times, removing each source in turn following the principle of superposition. This was done by providing a wrapper class known as M^{2}CIRQ for Multiple Source Multiple level CIRQ.
- Unlike electrical systems, the actual substances flowing in a circuit may be important in fluid flow systems to represent faults such as leaks where for

example air may enter the circuit and cause components to behave differently. To address this issue the vertices were extended by including a list of substances present (flowing into) at the vertex. These substances are then accessed by the edges to cause propagation according to the derived flows. A vertex therefore causes 'mixing' of the substances to occur based on the flows.

Both of these issues plus other more minor considerations such as the possibility that sources may provide more than one qualitative level of voltage or effort are detailed in [Snooke, 2007], describing continuing work carried out after the implementation in this work was completed.



F149	**At LH normal on (Step_3)**						RP2_4_RL_LH - leak.	8	8		64
		Observable	**Value**	**Function**	**Effects**	**S,D**					
		'OC_WT_LH.tank_level'	lower than expected (normal level decrease expected)	'engine_supply_left' failed	left engine fuel starvation	8, 8					
		'FT_FL_LH.flow'	high (normal expected)								
	At AT_LH on (Step_14)										
		Observable	**Value**	**Function**	**Effects**	**S,D**					
		'TK_AT_LH.tank_level'	lower than expected (normal_level_decrease expected)	'auxiliary_transfer_left' failed	left auxiliary fuel unusable	5, 5					
	At RH crossfeed on (Step_22)										
		Observable	**Value**	**Function**	**Effects**	**S,D**					
		'FT_FL_RH.flow'	high (normal expected)	'engine_crossfeed_to_right' failed	right engine fuel starvation	8, 8					
		'OC_WT_LH.tank_level'	lower than expected (normal level decrease expected)								
F150	**At RH normal on (Step_6)**						FL1_3_FS_RH - blocked.	8	8		64
		Observable	**Value**	**Function**	**Effects**	**S,D**					
		'FT_FL_RH.flow'	none (normal expected)	'engine_supply_right' failed	possible engine malfunction	8, 8					
		'PT_FL_RH.presssure'	none (pumpnominal expected)								
		'OC_WT_RH.tank_level'	no level change (normal level decrease expected)								
F151	**At RH normal on (Step_6)**						FL1_3_FS_RH - partialblocked.	8	8		64
		Observable	**Value**	**Function**	**Effects**	**S,D**					
		'FT_FL_RH.flow'	low (normal expected)	'engine_supply_right' failed	right engine fuel starvation	8, 8					
		'OC_WT_RH.tank_level'	higher than expected (normal level decrease expected)								
F152	**At RH normal on (Step_6)**						FL1_3_FS_RH - fracture.	8	8		64
		Observable	**Value**	**Function**	**Effects**	**S,D**					
		'FT_FL_RH.flow'	none (normal expected)	'engine_supply_right' failed	possible engine malfunction	8, 8					
		'OC_WT_RH.tank_level'	higher than expected (normal level decrease expected)								
	At LH crossfeed on (Step_19)										
		Observable	**Value**	**Function**	**Effects**	**S,D**					
		'OC_WT_RH.tank_level'	higher than expected (normal level decrease expected)	'engine_crossfeed_to_left' failed	possible engine malfunction	8, 8					
		'FT_FL_LH.flow'	none (normal expected)								

Figure 5.15: Fragment of FMEA generated from system in Figure 2.3

Chapter 6

Conclusions

It has been shown that the techniques presented here can resolve simple circuits that collapse to a set of single resistances that connect power and ground. In addition, the automatic generation of new qualitative levels works in test circuits and has been seen to help analysis in the ASTRAEA project, providing new qualitative levels in circuits, helping to identify ground loops where this would not have been possible with previous simulators.

The quantity space may be of more than the simple three values that have long been used by the commercial tools that were developed using early forms of qualitative circuit solving. The primary objective of this thesis — to allow automatic qualitative value generation in the *flow* quantity space where necessary — has been demonstrated to work experimentally.

There have been a number of enhancements to made to the algorithm as proposed in [Lee, 2002] that have allowed many more circuits to be solved. These include the removal of infinite branches, loops, floating branches, and the collapsing of zero-resistance edges to assist in resolving a lot of the problems that were being caused by bridges in real circuits. The system also only generates current values when they are queried therefore potentially reducing the total number of calculations needed as often only parts of an electrical circuit will be queried. This can assist in providing solutions to circuits that will only partially reduce, as long as the queried segment will reduce to a single resistance, then a solution can be provided.

In cases where the circuit does not collapse to a single resistance, by issuing a call-back to the calling program, which then will usually relay that on to the user, we can produce results that were previously unavailable or unreliable. These call-backs could be cached in an intermediate module, allowing for storage of known results, therefore not having to re-present already-seen cases to the user, and could be stored long term in a non-volatile knowledge base. There are a finite number of combinations and permutations that occur on a regular basis, and it should not take long to build up the requisite knowledge base. It is felt that this task is beyond the scope of this dissertation.

Some initial work has been done into the automatic resolution of the *bridge*, *delta* and *star* circuits, by development of a knowledge based approach, drawing from a library of examples and caching of user supplied knowledge. As this problem has been understood to be one of the difficulties encountered in the previous attempted implementations, there was some thought put into the design to allow easy modular expansion of the system in the future if necessary. The call-back system implemented for this has been used for this purpose, and has been found to work. The enhancements that have been made to the algorithm it has been found that this is not as necessary as it was thought to be initially, as all the real-world circuits presented to the system were able to be processed by the algorithm.

Code has been written to implement the new algorithms, and demonstrated to work in the electrical problem domain, through the development of a new tool, which has been used for some time within Aberystwyth University, and has been used on the ASTRAEA project in both the electrical and fluid flow domains. The product of this thesis and its integration into the larger system has made it possible to analyse engineering systems that were not able to be addressed by the existing circuit analysers, most importantly in systems that contain multiple sources and types of effort.

6.1 Limitations

The system allows the creation of multiple parallel edges between the same vertices, and if a name for an edge is not specified, and the edges have the same resistance, then there is no way of getting a handle for one or other edge in particular. This is potentially a problem, but if it is borne in mind when generating the circuit, and edges are given unique names should not actually prove to be a serious problem for the programmatic use of the system.

At the moment, the system has not been developed to consider the resolution of voltage, or pressure drop at points throughout the system being simulated. The primary goal for the simulation has been for determination of flow or current levels. In an ideal simulation environment, the system should be able to determine pressure differential between a point in the circuit and the source and/or sink point.

6.2 Future Work

I would like to perform some complexity analysis on the code in order to determine where optimisations could be applied, so that the scalability of the implementation can be determined. This was not possible in the scope of this piece of work due to time constraints. Experience of using the system against previous systems implies that this new algorithm and implementation performs simulations in less time, and provides more reliable results than were provided by previous software systems.

If a more reliable way than the knowledge based approach of collapsing resistance networks and resolving current flows through bridge, star and delta configurations was available, that could be added to this program through the call-back mechanism - with the programmer not having to modify the central core of the system.

Bibliography

[*ASTRAEA*] http://www.projectastraea.co.uk/the-programme/

[Arnborg et al., 1991] Arnborg, S., Courcelle, B., Proskurowski, A., and Seese, D. (1991). An algebraic theory of graph reduction. In Ehrig, H., Kreowski, H.-J., and Rozenberg, G., editors, *Graph Grammars and Their Application to Computer Science*, volume 532, pages 70–83. Lecture Notes in Computer Science.

[Bénazéra and Travé-Massuyès, 2003] Bénazéra, E. and Travé-Massuyès, L. (2003) The consistency approach to the on-line prediction of hybrid system configurations, IFAC Conference on Analysis and Design of Hybrid Systems (ADHS'03), Saint-Malo (France).

[Bobrow, 1985] Bobrow, D. (ed), *Qualitative Reasoning About Physical Systems*, North-Holland, 1985 (originally published as Artificial Intelligence vol. 24, 1984).

[Dijkstra, 1959] Dijkstra, E. W. (1959). A note on two problems in connection with graphs. *Numerische Math.*, pages 269–271.

[de Kleer, 1984] de Kleer, J. (1984). How Circuits Work. *Artificial Intelligence* 24, pages 205-280.

[de Kleer and Brown, 1984] de Kleer, J., and Brown, J. (1984). A Qualitative Physics Based on Confluences, *Artificial Intelligence*, 24, pages 7–83.

[Downes, 2007] , Downes, C. (2007) Astraea T7: an architectural outline for system health management on civil UAVs. In Proceedings 2nd Autonomous Systems Conference, IET, November 2007

[Forbus and Falkenheiner, 1991] Forbus, K. and Falkenheiner, B. (1991) Compositional modelling: finding the right model for the job. Artificial Intelligence, vol 51, pages 95-143.

[Hunt et al., 1993] Hunt, J.E., Lee M.H., Price C. J. (1993) Automating the FMEA Process, International Journal of Intelligent Systems Engineering, vol 2(2), pp119-132.

[Lee, 1999] Lee, M. H. (1999). Qualitative circuit models in failure analysis reasoning. *AI Journal*, 111:239–276.

[Lee, 2000] Lee, M. H. (2000). Qualitative modelling of linear networks in engineering applications. In *Proceedings of ECAI-2000*, Berlin, pages 161–165.

[Lee, 2002] Lee, M. H. (2002). Structural and parametric change in circuit modelling using CIRQ and MCIRQ. Internal memo, Department of Computer Science, University of Wales, Aberystwth. Also Appendix A to this thesis.

[Lee and Ormsby, 1992] Lee, M. H. and Ormsby, A. R. T. (1992). Qualitative modelling of electrical circuits. In *Proceedings of the 6th International Workshop on Qualitative Reasoning about Physical Systems*, pages 155–169. (QR'92) Department of Electrical Engineering, Heriot-Watt University, Edinburgh.

[Lee and Ormsby, 1993] Lee, M. H. and Ormsby, A. R. T. (1993). Qualitiatively modelling the effects of electrical circuit faults. *Artificial Intelligence in Engineering*, 8:293–300.

[Lee et al., 2001] Lee, M. H., Bell, J., and Coghill, G. M. (2001). Ambiguities and deviations in qualitative circuit analysis. In *Proceedings of the 15th International Workshop on Qualitative Reasoning:*, pages 51–58. QR'01, San Antonio, Texas, May 17-18.

[Mauss and Neumann, 1996] Mauss, J. and Neumann, B. (1996). Qualitative reasoning about electrical circuits using series-parallel-star trees. In *Proceedings of the 10th International Workshop on Qualitative Reasoning in Physical Systems*, volume QR'96, pages 147–153.

[Mentor Graphics website, 2011] Mentor Graphics Capital Harness Website, showing Design Analysis tools, visited 2nd June 2011. http://www.mentor.com/solutions/aerospace/electrical-systems-design-and-harness-engineering/.

[Milne, 1991] Milne R. (1991) Model-based reasoning: The applications gap. *IEEE Expert 6(6)*: pages 5–7.

[Price, 1998] Price, C. (1998) Function Directed Electrical Design Analysis, Artificial Intelligence in Engineering 12(4), pages 445-456.

[Price, 2000] Price C.J. (2000) AutoSteve: Automated Electrical Design Analysis, in Prestigious Applications of Artificial Intelligence (PAIS-2000), Berlin, in Proceedings ECAI-2000, pages 721–725.

[Price and Lee, 1988] Price, C.J. and Lee, M.H., (1988) Applications of Deep Knowledge, *Artificial Intelligence in Engineering*, pp12-17, volume 3(1).

[Price and Taylor, 2002] Price C. and Taylor, N. (2002) Automated Multiple Failure FMEA, Reliability Engineering and System Safety 76(1), pages 1–10.

[Price et al., 1996] Price, C. J., Snooke, N., Landry, J. (1996) Automated Sneak Identification, Engineering Applications of Artificial Intelligence, volume 9(4), pages 423–427.

[Price et al., 1997] Price, C. J., Pugh, D. R., Hunt, J. E., and Wilson, M. S. (1997). Combining functional and structural reasoning for safety analysis of electrical designs. *Knowledge Engineering Review*, 12(3):271–287.

[Price et al., 1999] Price, C. J., Snooke, N., and Ellis, D. (1999). Identifying design glitches through automated design analysis. In *Innovative CAE Track, Proc. Ann. Reliability and Maintainability Symp.*

[Price et al., 2003] Price, C. J., Snooke, N., and Lewis, S. (2003). A layered approach to automated electrical safety analysis in automotive environments. *Computers in Industry* vol 57, pages 451–461.

[Price et al., 2006] Price C.J., Travé-Massuyès L., Milne R., Ironi L., Forbus K., Bredeweg B., Lee M., Struss P., Snooke N., Lucas P., Cavazza M., Coghill G.M. (2006) Qualitative Futures. *Knowledge Engineering Review*, vol 21(4), pp317–334.

[Savakoor, 1993] Savakoor, S., Bowles, J. R., Bonnell, (1993) D. Combining sneak circuit analysis and failure modes and effects analysis, in Proceedings of Annual Reliability and Maintainability Symposium, pages 199–205, IEEE Press.

[Snooke, 1999] Snooke, N. A. (1999). Simulating electrical devices with complex behaviour. *AI communications special issue on model based reasoning*, 12(1-2):44–59.

[Snooke, 2007] Snooke, N. (2007). M^2CIRQ: Qualitative fluid flow modelling for aerospace FMEA applications. In Price, C. J., editor, *21st International Workshop on Qualitative Reasoning*, pages 161–169. Aberystwyth University.

[Snooke and Price, 2011] Snooke N. and Price C. (2011). An Effective Practical Model-Based Detectability Tool. Submitted to DX-2011.

[Snooke and Shipman, 2001] Snooke, N. A. and Shipman, R. C. (2001). Generating automotive electrical system models from component based qualitative simulation. In *Research and Development in Intelligent Systems*, volume XVIII, pages 100–114. Springer.

[Snooke et al., 2008] Snooke, N., Price, C., Downes, C., and Aspey, C. (2008). Automated failure effect analysis for PHM of UAV. In *Proceedings of the International System Safety and Reliability Conference.* ISSRC.

[Struss and Price, 2003] Struss P. and Price C.J. (2003) Model-based systems in the automotive industry, AI Magazine special issue on Qualitative Reasoning, pp17-34, vol. 24(4).

[Struss et al., 1995] Struss, P., Malik, A., Sachenbacher, M. (1995). Qualitative Modeling is the Key. In *Proceedings of the 6th International Workshop on Principles of Diagnosis (DX-95)*, Goslar, Germany.

[Travé-Massuyès and Milne 1995] Travé-Massuyès, L. and Milne, R. (1995) Application-oriented qualitative reasoning. Knowledge Engineering Review, 10(2) pages181–204.

[Travé-Massuyès and Milne 1997] Travé-Massuyès, L. and Milne, R. (1997) Gas-Turbine Condition Monitoring Using Qualitative Model-Based Diagnosis. IEEE Expert 12(3): pages 22–31.

[Ward and Price, 2001] Ward, D. and Price, C. (2001) System functional safety through automated electrical design analysis, SAE 2001 Transactions, Journal of Passenger Cars, Section 7, vol 110: Electronic and electrical systems, pages 341-347.

[Williams and Nayak, 1996] Williams B., Nayak P. (1996) A Model-based Approach to Reactive Self-Configuring Systems, Proceedings of AAAI-96, Portland, Oregon, pages 971–978.

[Wotawa, 1999] Wotawa, F. (1999). Guest–Editorial Special Issue on Model-Based Reasoning. *AI Communications*, 12(1/2).

Appendix A

Structural and Parametric Change in Circuit Modelling using CIRQ and MCIRQ by M.H. Lee

Dougal/SoftFMEA project

Internal Memo

17 October 2002

Structural and Parametric Change in Circuit Modelling using CIRQ and MCIRQ

Mark H. Lee
Department of Computer Science
University of Wales, Aberystwyth, UK
mhl@aber.ac.uk

The theory behind the qualitative electrical circuit modelling methods known as CIRQ and MCIRQ is directly based on standard electrical analysis. Although qualitative models are abstractions, and therefore will lack information that may or may not be important, they should not perform any differently from the ways that the standard theory predicts. However there will be many ways to implement any theory and variations between algorithms will cause performance differences. A big problem with qualitative methods is that implementations can inadvertently misrepresent the theory in some situations. This is because the theories are so abstract that they often give no indications about how implementations should proceed and even outline algorithms are often not available or fully developed.

This note attempts to clarify the fundamental theory of MCIRQ in order that implementations can faithfully follow the method and produce results that are correct in that they match those that would be produced by standard circuit theory. MCIRQ will be used throughout as this will subsume the same issues in CIRQ.

1. DUALITY

The principle of duality states that every electrical state or situation has a dual state that has exactly the same structure, state or characteristics, provided that certain key variables are mapped between the duals.
This is very useful because when we have one result, say for a series circuit, we can check or deduce the result for the dual situation, in this case a parallel circuit.

MCIRQ finds the activity levels in the branches of a circuit by reducing down any series and parallel branches to simpler equivalent circuits and finding the current flow in the reduced case, Then the reduced branches are expanded back to their original configuration and during this process the currents are assigned as appropriate.

First consider some simple series and parallel circuits and how they would be solved using standard electrical theory.

1.1 Series Circuits

Consider 3 resistors in series, of values 10, 100, and 1000 ohms. Let there be a voltage of 10 volts applied.

The total resistance is the sum of the branches = 1110 ohms and the current flow is V/R = 10/1110 = 9 milliamps. The voltage over each resistor can now be calculated from V = I * R.

Notice that he higher the value of a resistor as a proportion of the total, the more of the supply voltage will exist across it, (see figure 1). In fact, $V_R = V_{supply} * R/\Sigma r$ for a particular resistor R.

1.2 Parallel Circuits

Consider 3 resistors in parallel, of values 10, 100, and 1000 ohms. Again a voltage of 10 volts is applied.

The total resistance, R, of three resistors, A, B and C, in parallel is given by:

$$R = (A*B*C)/(A*B + B*C + A*C)$$

For this case, R = 1000/111 = 9 ohms. The current flow in each resistor can be calculated from I = V/R, this gives 1.11 amps for the total circuit and 1 amp, 100 milliamps, and 10 milliamps for the individual branches (see figure 1).

We see that the lower the resistance of a branch in a parallel circuit the more of the total current it will carry.

It is worth noting another dualality, that is not always observed. In series circuits across the power supply terminals, we usually assume that the type of supply is constant voltage (regardless of the current drawn). This is idealised, but if we do assume this then we should also consider that a parallel circuit across the supply will see the power source as a constant current source. This would prevent infinite current being drawn in power short circuit conditions.

1.3 Qualitative Circuits

We now examine qualitative versions of the above circuits. Figure 2 shows the same circuits with qualitative resistance values assigned. The rules for series/parallel reduction state that a series branch is equivalent to a single resistance of value Max(r) for all values r, and a parallel branch is equivalent to a single resistance of value Min(r) for all values r.
Thus, the given series circuit in figure 2 has an equivalent value of "hi". Also, most of the supply voltage will appear across the single "hi" component. The ordering of the qualitative values is based on an order of magnitude assumption, that is "lo" << "med" << "hi," and hence the others are **much** smaller than "hi". In general, because any non-dominant components will have **much less** influence, they can often be **neglected**. Notice that if there were two or more resistors of the same value "hi" then they would share the supply voltage pro rata between them. We can mark any component(s) that dominate in this way.

The parallel circuit shows a similar effect. The circuit is equivalent to a single "lo" value and the main flow path(s) can be traced and marked, see figure 2. Again, there will be some flow in the non-dominant branches but our qualitative algebra allows us to **neglect** these if we wish.

2. PARAMETRIC VERSUS STRUCTURAL CHANGES

In theory, the extremal values of resistance, 0 and ∞, are no different from any other values. This is particularly true in numerical analysis and the equation solving process will not be affected in any way by any actual circuit parameter values, (apart from taking care with numerical procedures, e.g. to avoid divide by zero etc.). For example, if the 1000 ohm resistor in figure 1 was changed to ∞ then, in the series circuit, the current would be zero and all the supply voltage would be across the (open-circuit) resistor, while in the parallel circuit only one branch would have zero current with the rest being unchanged. Alternatively, if we change the 10 ohm resistor to zero then, in the parallel circuit all the current would flow through the zero valued resistor and all the other branches would have no current (shorted-out), while in the series circuit just that one branch would have zero voltage across it with the rest being relatively unchanged.

The reason why the numerical approach works so well is that the structure of the system does not change, just the **parameters** (resistance values) are altered. However, there is a **qualitative difference** between **some** current flow through the minor branches or **some** voltage across the lesser components and **no** current or **no** voltage **at all**. In most realistic situations, resistance must always be non-zero however small and therefore in our parallel circuit there will always be some flow in all the branches, however small.

But zero resistance can also indicate a **short-circuit** condition, that is, a zero resistance wire placed between two terminals joins them together so that anything previously connected between them now forms a closed loop that by definition must be electrically inactive. Thus, in the parallel case in figure 3 the terminals T_1 and T_2 are effectively joined by the short-circuit condition, that is T_1 and T_2 are **electrically the same terminal** and the other branches connected to them are therefore dead. This is a **structural** change that needs to be captured by any qualitative formulation. Similarly, for a series circuit an **open-circuit** is a dual form that also causes dead branches.

In general, structural changes can not be handled by numerical methods but the results are correct for zero and infinite resistance cases. If true zero and infinite values are to be used in qualitative models then we must recognize these as different from the non-extremal values and treat them accordingly.

To deal with the structural changes produced by the two extremal parametric cases we can employ two graph-theoretic concepts:

- Supernodes
- Dangling edges

Both of these have been described and used before in qualitative algorithms.

Supernodes are a way of joining together terminals that are electrically identical. A supernode is a cluster of terminals formed by linking all the terminal names. This provides the representation necessary to deal with short-circuit cases. Supernodes can be most simply implemented as lists of terminals.

Dangling edges are circuit segments that do not have two non-intersecting paths to two separate supply terminals, and are therefore, by definition, electrically dead. By using spanning trees over the circuit any dangling sections can be eliminated. An alternative and simpler way of removing dangling edges is to remove any edges leading to nodes of degree 1.

3. IMPLEMENTATION OF QUALITATIVE PARAMETRIC AND STRUCTURAL CHANGE

The design of MCIRQ is based around the reduction and expansion of series/parallel (SP) circuits. SP circuits do not contain bridges and account for the great majority of practical electrical circuits. After reduction an SP circuit will give a single equivalent resistance value from which the total circuit current can be computed. The circuit is then expanded and the branch current values are assigned at each expansion step by using knowledge of the total current. Any non-SP circuits are also first reduced to remove all SP parts and the only difference is that, rather than a single resistance, a small compressed circuit is produced and currents have to be assigned by using a separate technique.

A key feature of the method is the segment table that holds data on each series or parallel branch both before and after compression. This stores the equivalence between all the reduction stages and allows expansion.
The segment table is also a repository for all relevant data on the circuit sections, including numeric and qualitative values.

It is worth illustrating the process with a worked example. Figure 4 shows a network of resistors, each with one of 5 qualitative values.

3.1 COMPRESSION of circuit branches

Compression starts with series branch removal - a search of the network looks for all nodes of degree 2 (i.e. with 2 edges) and replaces the two edges with an equivalent. Thus the node is removed and (as Max is used for series resistance combinations) the larger resistance remains. This is repeated until no more reductions occur. Figure 5 shows the result of the first S step (series-1). The next step is to reduce parallel branches. Any two or more edges that join the same two nodes can be replaced by a single edge of value equal to the minimum edge resistance. A search for all such cases thus removes parallel branches (step P 1), and figure 6 shows the result for our example circuit. Notice that the branch with three resistors in parallel has a zero value and so the nodes will be marked (see below).

Notice that removing one type of branch often leaves behind new cases of the other type, this is why S and P reduction stages must follow each other repeatedly until no further reduction occurs. So we now perform an S reduce again and reach the result shown in figure 7. Notice that any infinity valued edges are eliminated, this deals with open circuit conditions by producing dangling edges. Next follows another P reduction, giving figure 8. Then another S reduction gives figure 9. Another P reduction removes the last parallel branch, seen in figure 10, and a final S stage produces the final irreducible result seen in figure 11.

3.2 Current assignment

If SP reduction results in a single equivalent resistance value then the current is easily computed from a simple inverse relationship. For a resistance across a fixed voltage supply, Ohm's law states that the current will vary inversely with the resistance value. A table gives the relationship for qualitative values:

Resistance =	0	lo	med	hi	∞
Current =	M	hi	med	lo	0

where M is some maximum limit.

I+ I i-

In order to label our diagrams we will not use the same names for current but use **I** for med, **I+** for hi, and **i-** for lo current values. Figure 11 shows the current assignments: the main current is I, for medium levels, and the dangling edges have zero current.

3.3 Processing of open-circuit (dangling) edges

During the stages of SP reduction, any dead ends will eventually turn into single dangling edges attached to nodes of degree 1. When these are discovered they can be marked in the segment table and then ignored until compression has finished. During current assignment it is a simple matter to assign zero current to all marked dangling edges and then during expansion all their component parts will then be returned as electrically inactive.

3.4 EXPANSION back to original circuit

We now know the total current drawn by the whole circuit and the expansion stage can use this value to assign currents to all the branches and sub-branches in the original circuit. Any given branch will carry either the total value or something less than the total value, as determined by its own value and its position in the system. There will be at least one path through the whole circuit that flows at the total current value.

The segment table guides the whole process of expansion by reversing the compression sequence. Following the table in reverse order the single resistor is replaced by the previous two in series, then the last parallel reduction stage is reversed, causing two branches to replace the upper resistance. The circuit now looks like figure 12. At each step current is assigned in proportion to the resistance of the branch being reinstated. Thus a medium (or less) branch will receive a value of **I** while a hi branch will receive a value of **i-**. This gives the branch currents as shown in figure 12.

The next step is an S expansion, putting back several series branches. This does not change the topology of the circuit (or the currents) - it just adds nodes back - thus moving backwards from the state in figure 9 to that in figure 8. This gives the situation shown in figure 13. Next a P reversal stage then produces figure 14, with new branches being assigned their currents, as computed from the currents found in the previous stage. Another S reversal then follows, see figure 15, and this also restores some of the edges in the open-circuit part of the network (this automatically occurs at this point because this is the stage where it was compressed). Figure 16 shows the result of the next P reversal. Finally, an S reversal reaches the original circuit, seen in figure 17. This shows the parts that have zero current (dashed) and the (single) main flow path through the circuit (arrows); plain edges carry currents less than that associated with a resistance of med.

3.5 Implementation of supernodes in the segment table

During compression, if any parallel segment has a total resistance of zero then its two terminal nodes are marked as supernodes and added to any relevant existing supernode list. This is so that any shorted-out sections can be identified. On expansion, any such marked segment is labelled as follows: all zero resistance edges are assigned currents from the previous level, and all other (non-zero) edges are assigned zero current (electrically inactive). An example is seen in the resistor at the bottom left of figure 4: as this is in parallel with two other resistors it shorts them out and steals all the current. During compression the two terminals either side of the zero edge are noted and then during expansion currents are assigned accordingly. Figure 16 shows how an expansion step has assigned zero current to the other branches.

4. CURRENT ASSIGNMENT FOR NON-SP CIRCUITS

The remaining task is to solve the currents when SP reduction does not return a single resistance but delivers a small network. By definition **such networks will contain no nodes of degree less than 3 and no loops.** These are circuits with bridge edges that may be ambiguous in that they may be balanced with no current flowing or current may flow in either direction. Without numeric quantities it can sometimes be impossible to resolve this and in those cases such edges have to be labelled as **ambiguous**.

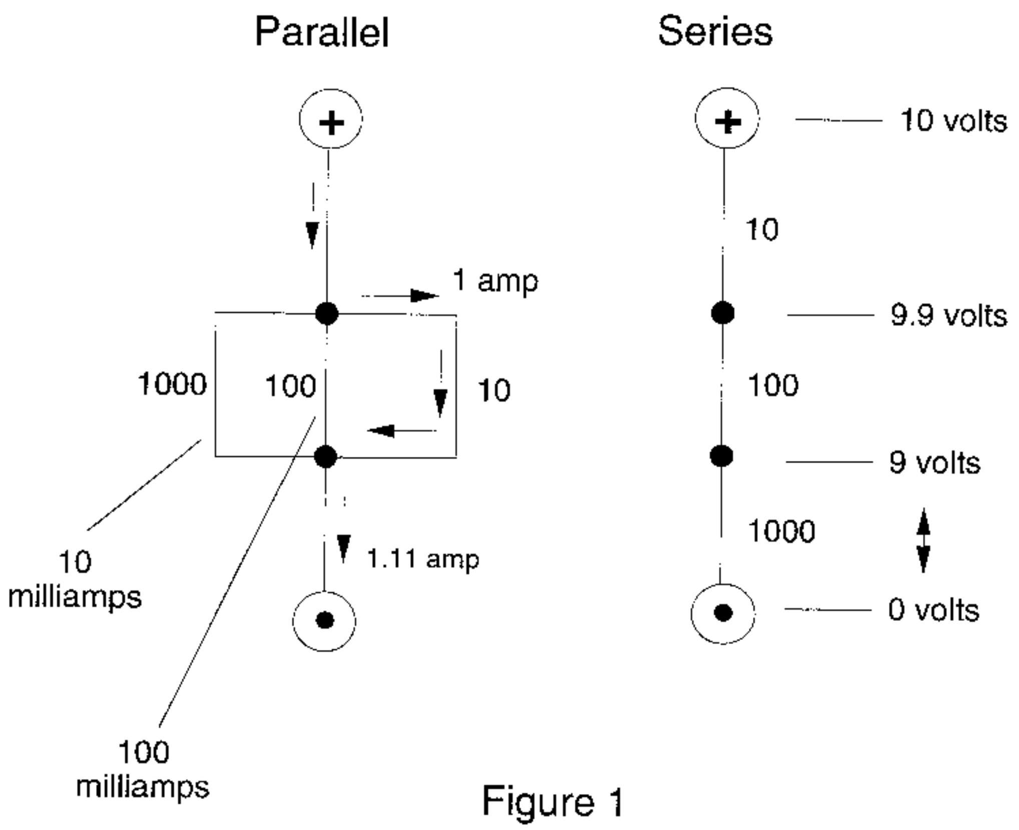
The standard method to analyse non-SP networks is to use Star/Delta transforms. These convert structures into different forms which can then be solved with further SP reduction. Figure 18 shows the relationship between three terminals using qualitative resistance values. The star form has an unused internal terminal but both are equivalent, that is, **they present the same resistance values as seen from the circuit outside**.

The relationship between the internal resistances of an equivalent star and delta are shown in the three equations. These use qualitative values and so addition must obey the order of magnitude constraint.

An example using the 5 edge diamond with bridge is shown in figure 19. This is the simplest non-SP circuit. The power supply is assumed to be across the top and bottom terminals. The first step is to locate a delta and transform into a star. Notice that a delta detection algorithm can look for the loop structure of 3 nodes and 3 edges. The values for the star are computed from the equations and then normal SP reduction can be applied. In this case the circuit reduces completely and the currents can be solved, see figure 20. We now know the currents entering or leaving the star terminals and these must be the same for the delta, hence we next replace the delta form as shown in figure 21. The resistance values of the delta arms combined with the flows required at the terminals can be used to deduce the internal currents. In this case the main current flows down the right hand side and a lesser current flows down the left hand side. The left current can not be of medium level as it can not flow out of terminal T2 and can not flow through the bridge. There is no indication of flow in either direction in the bridge and therefore this must be labelled as ambiguous.

Several star-delta transforms may be necessary to solve a particular circuit but they will eventually allow SP reduction and then the currents may be deduced. The rules of qualitative arithmetic must be applied when solving all circuits, for example, med + med = med, and a current through a branch in parallel with an order of magnitude lower path must itself be an order of magnitude lower than the other current.

Star-delta transforms are 3 node transforms; they can be extended into 4, node, 5 node and higher versions.



Parallel
Series
+
1 amp
1000
100
10
10 milliamps
100 milliamps
1.11 amp
10 volts
10
9.9 volts
100
9 volts
1000
0 volts

Figure 1

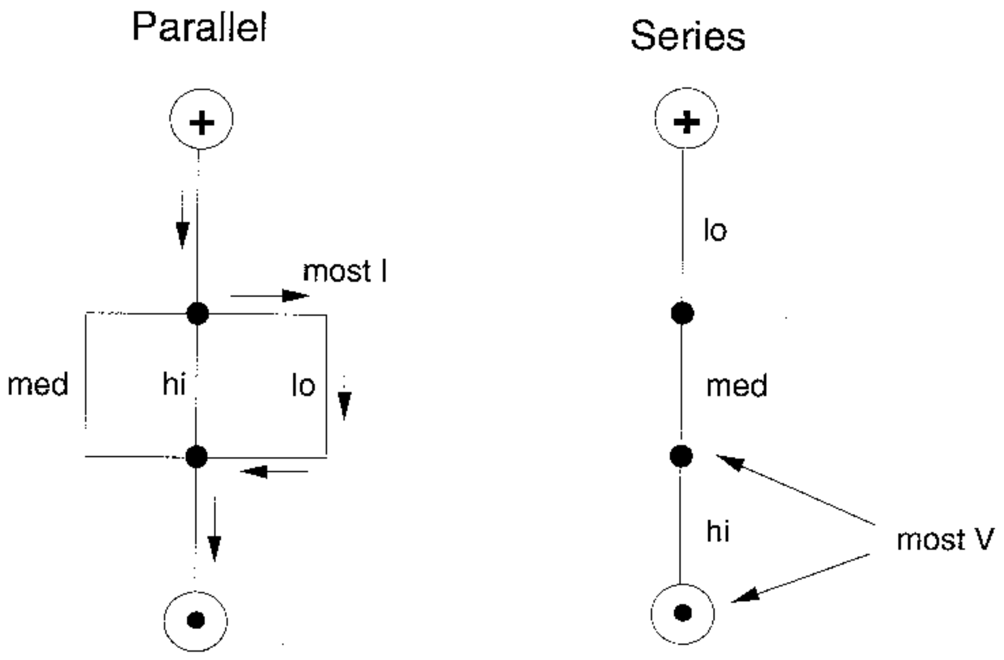


Figure 2

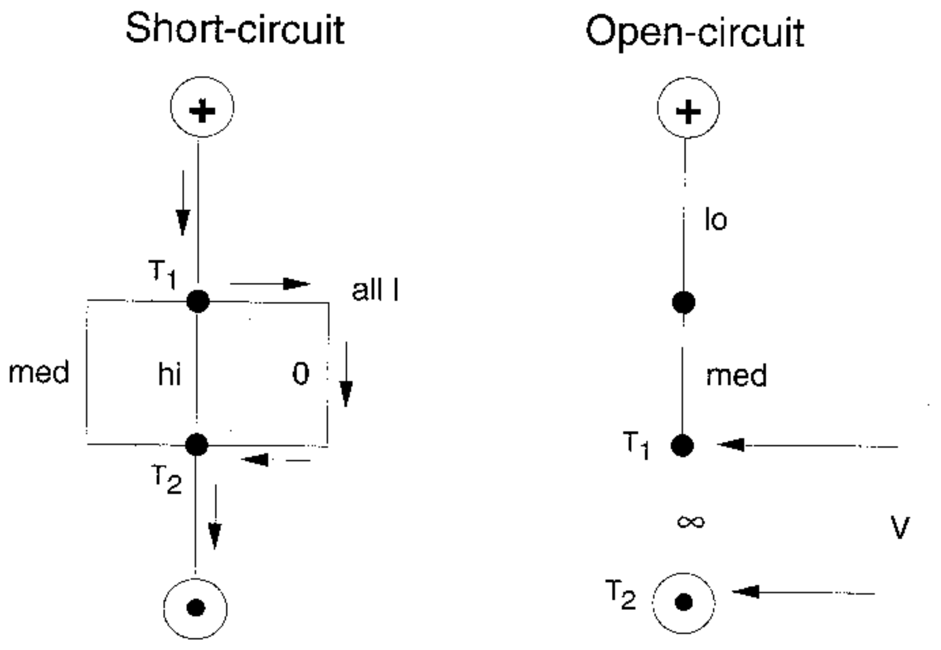


Figure 3

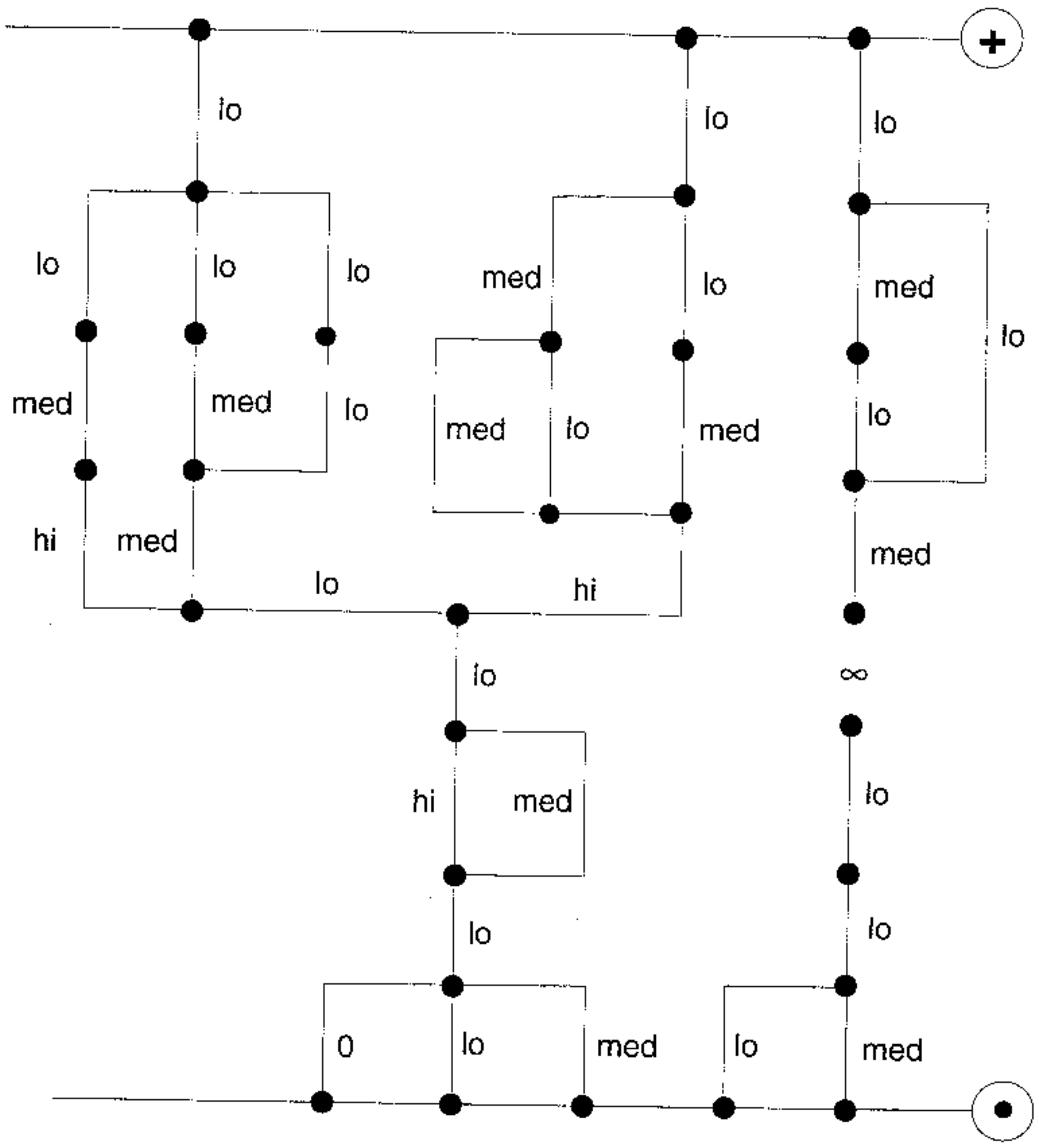

+
lo
lo
lo
lo
lo
lo
med
lo
med
med
lo
lo
med
med
lo
med
lo
med
lo
hi
med
lo
hi
med
lo
∞
hi
med
lo
lo
lo
0
lo
med
lo
med

Figure 4

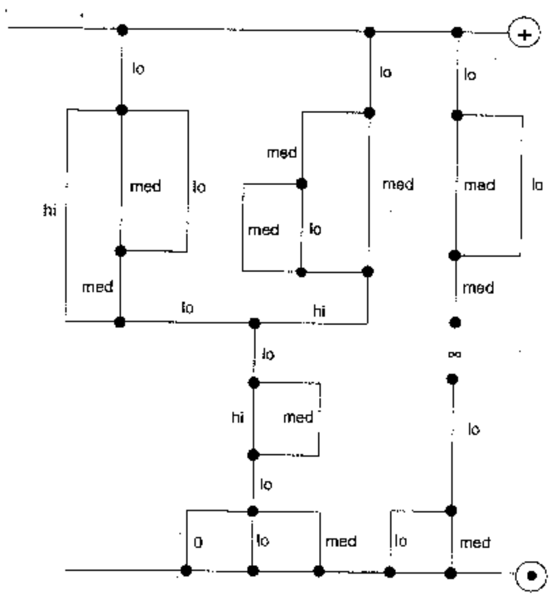


Figure 5

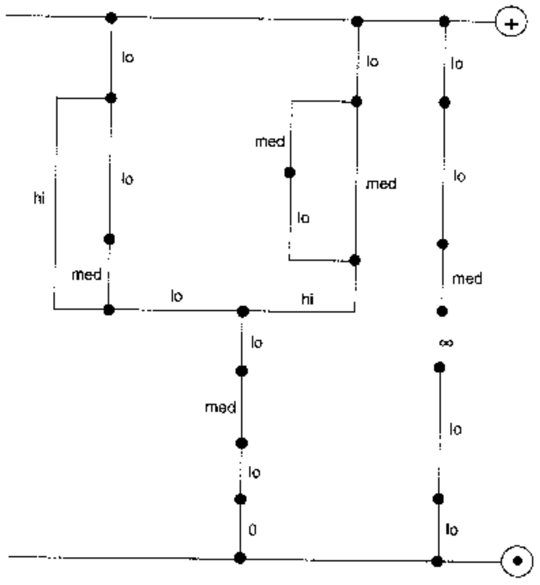


Figure 6

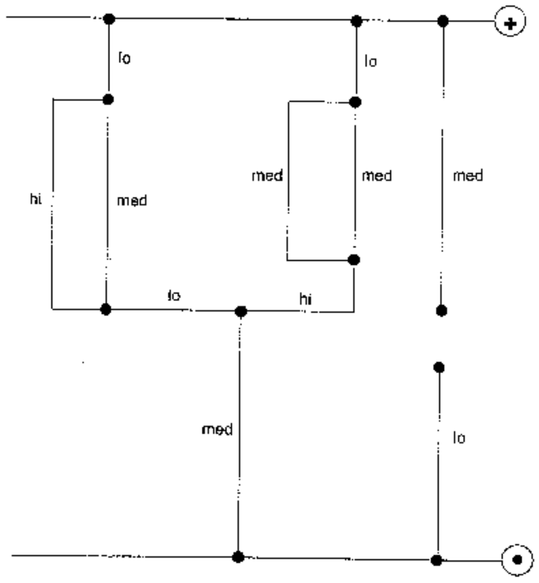


Figure 7

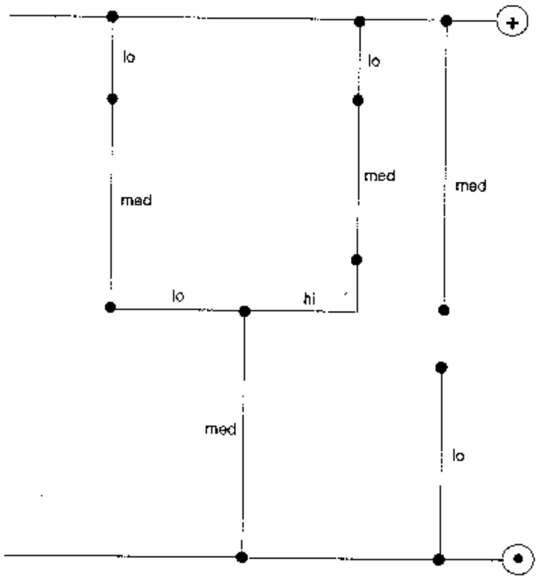


Figure 8

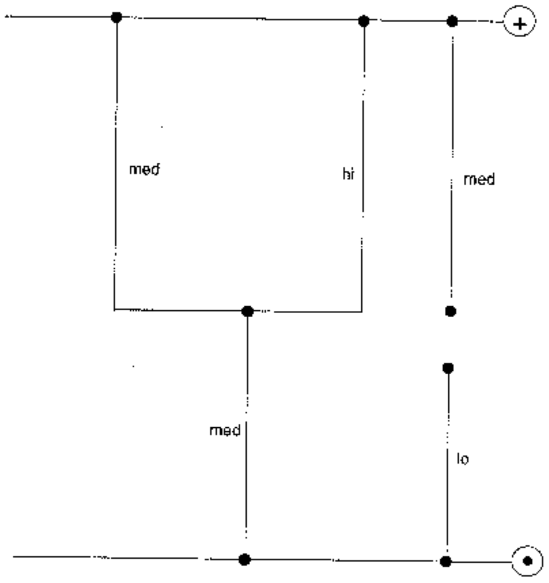


Figure 9

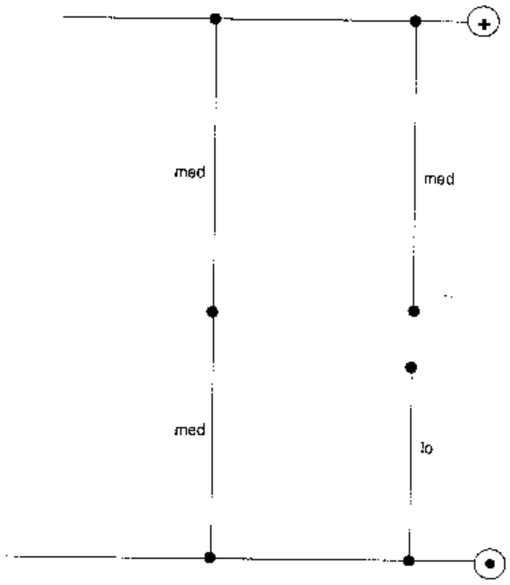


Figure 10

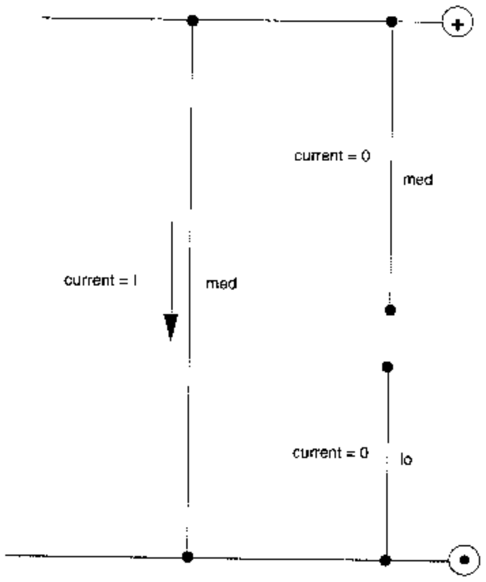


Figure 11

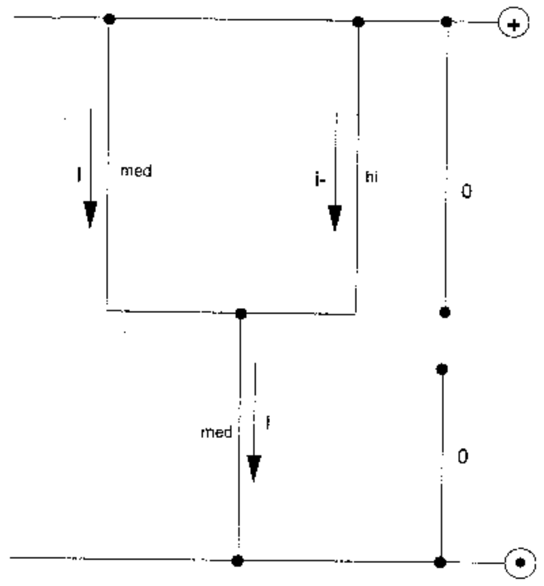


Figure 12

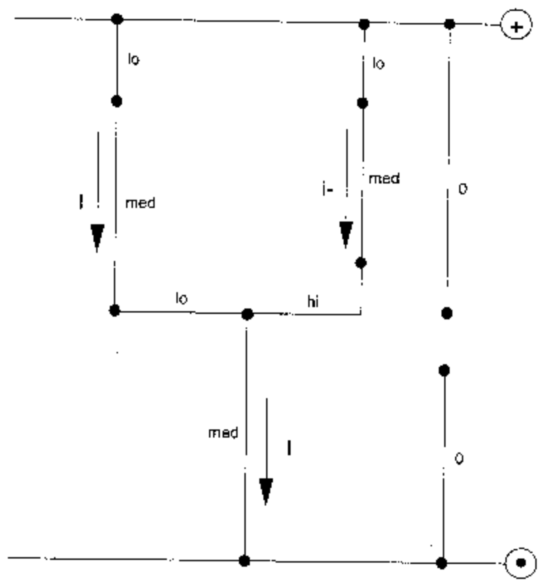


Figure 13

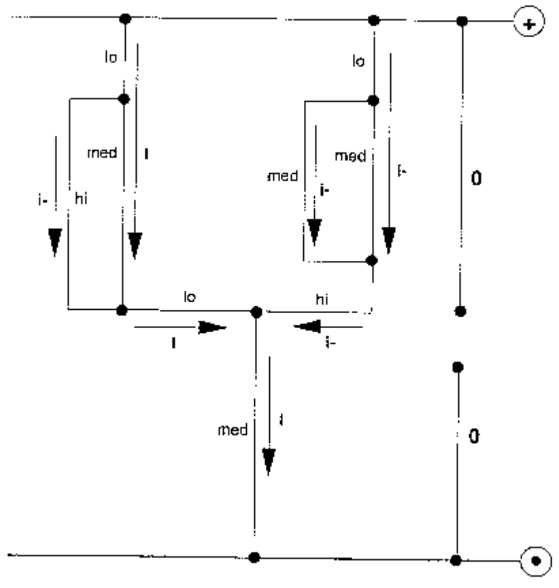


Figure 14

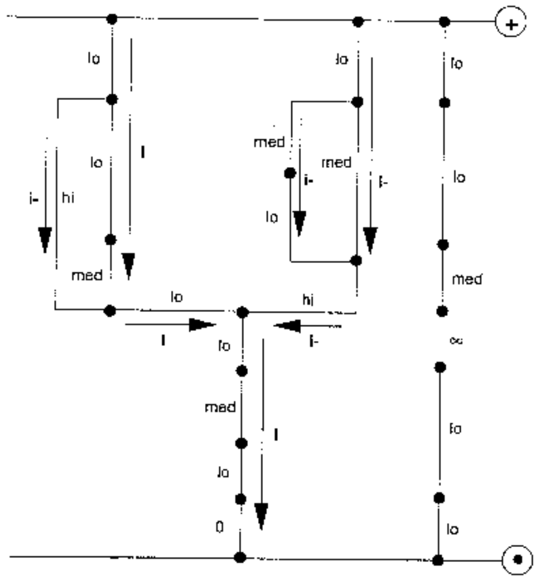


Figure 15

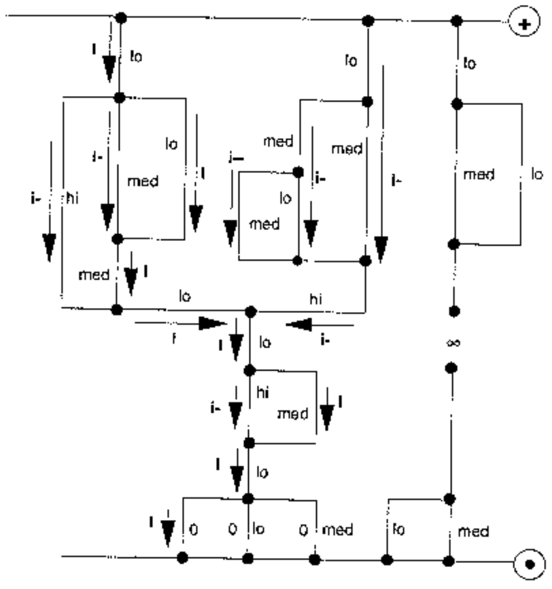


Figure 16

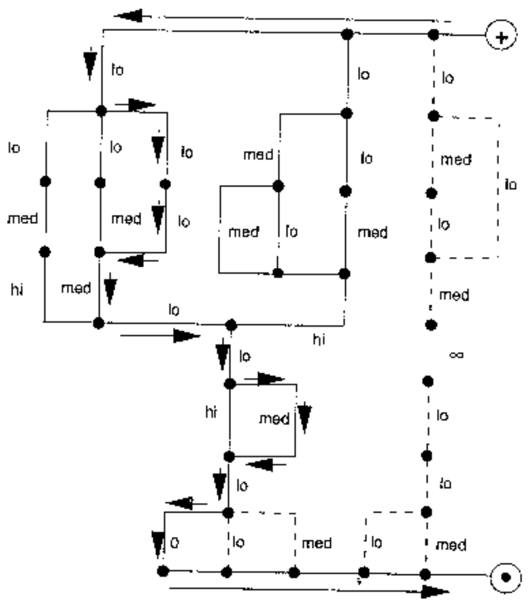
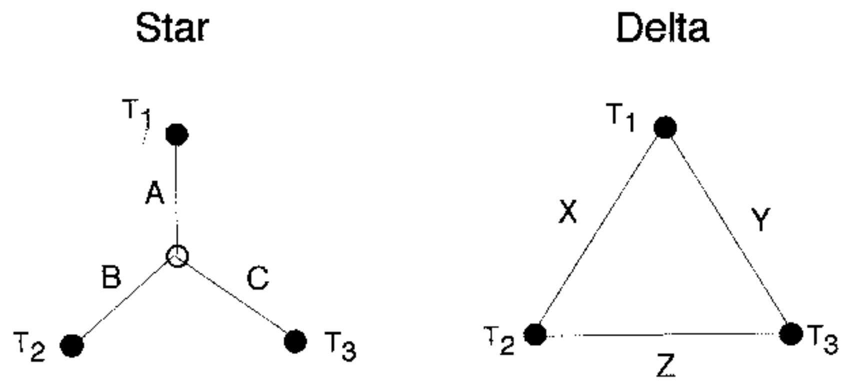


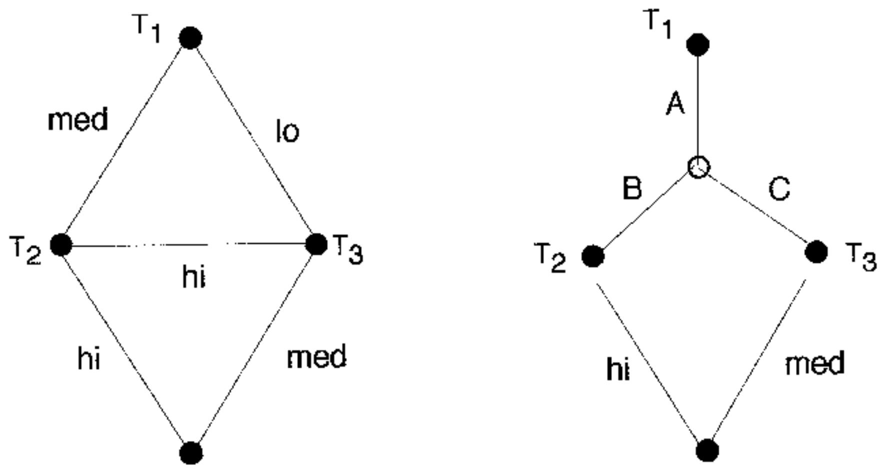
Figure 17



$$\begin{aligned} A + B &= \min(X,\ Y + Z) \\ A + C &= \min(Y,\ X + Z) \\ B + C &= \min(Z,\ X + Y) \end{aligned}$$

Figure 18

An example non-SP circuit



A + B = min(med, lo+hi) = med
A + C = min(lo, med+hi) = lo
B + C = min(hi, med+lo) = med

So A = lo
C = lo
B = med

Figure 19

Now solve by SP reduction:

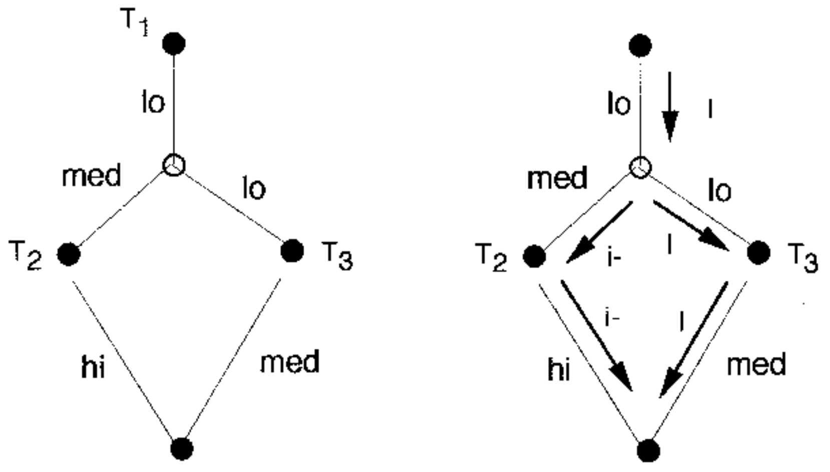


Figure 20

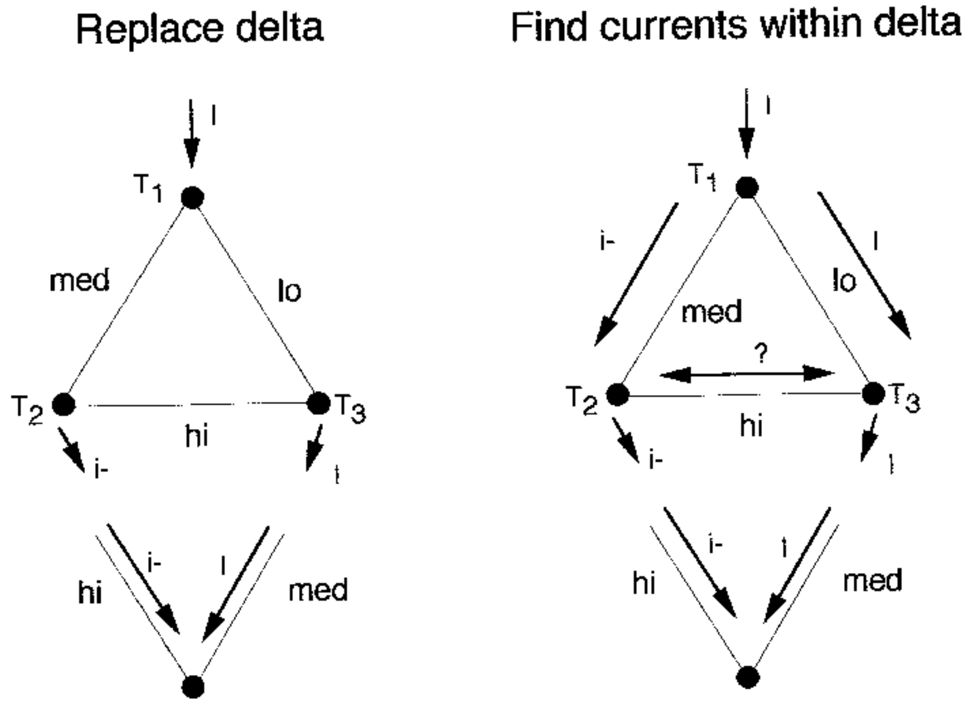

Replace delta
Find currents within delta
I
T1
T2
T3
med
lo
hi
i-
I
i-
I
hi
med
I
T1
i-
I
med
lo
?
T2
T3
hi
i-
I
i-
I
hi
med

Figure 21