A THEORY OF JUSTIFIED REFORMULATIONS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

By
Devika Subramanian
March 1989

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Michael R. Genesereth
(Principal Advisor)

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Bruce G. Buchanan

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Nils J. Nilsson

Approved for the University Committee on Graduate Studies:

Dean of Graduate Studies

Abstract

Present day systems, intelligent or otherwise, are limited by the conceptualizations of the world given to them by their designers. This thesis explores issues in the construction of adaptive systems that can incrementally reformulate their conceptualizations to achieve computational efficiency or descriptional adequacy. A detailed account of a special case of the reformulation problem is presented: we reconceptualize a knowledge base in terms of new abstract objects and relations in order to make the computation of a given class of queries more efficient.

Automatic reformulation will not be possible unless a reformulator can justify a shift in conceptualization. We present a new class of meta-theoretical justifications for a reformulation, called irrelevance explanations. A logical irrelevance explanation proves that certain distinctions made in the formulation are not *necessary* for the computation of a given class of problems. A computational irrelevance explanation proves that some distinctions are not *useful* with respect to a given problem solver for a given class of problems. Inefficient formulations make irrelevant distinctions and the *irrelevance principle* logically minimizes a formulation by removing all facts and distinctions in it that are not needed for the specified goals. The automation of the irrelevance principle is demonstrated with the generation of abstractions from first principles. We also describe the implementation of an irrelevance reformulator and outline experimental results that confirm our theory.

Acknowledgements

I would like to thank

- First and foremost, my advisors Professors Genesereth, Buchanan and Nilsson who have inspired me to do AI. Professor Genesereth suggested the problem solved in this thesis; he and Professor Nilsson constantly pushed me towards a rigorous formulation of both the problem and the solution. Professor Buchanan led me through the vast world of philosophical literature and impressed upon me the need for empirical validation of AI theories.

- My colleagues and co-conspirators in AI – David Chapman, Tom Dietterich, Jeff Finger, Russell Greiner, John Lamping, Jock Mackinlay, Patricia Riddle, David Smith, and Steve Tappel.

- My mentors and teachers – Gordon Bower, Rod Brooks, Lindley Darden, Cordell Green, Barbara Hayes-Roth, Doug Lenat, John McCarthy, Marvin Minsky, Tom Mitchell, Jack Mostow, Susan Owicki, Vaughan Pratt, Edwina Rissland, Paul Rosenbloom, Gerry Sussman, Richard Waldinger and Pat Winston.

- IBM for supporting me through my last four years at Stanford.

- My friends who made my sojourn through Stanford a truly memorable one – Leonor Abraido-Fandino, Elaine Bradshaw, Chris Fraley, Benjamin Grosof, Mike and Noi Hewett, Haym Hirsh, Jane Hsu, Anna Karlin, Peter Karp, Mei Lu, Pandu Nayak, Eunok Paek, Stuart Russell, Narinder Singh, David Wilkins, Marianne Winslett, Liz Wolf, John Woodfill, and Ramin Zabih.

- The Indian Junta for all the good times and the good food: Julie Basu, Nita Goyal, Akhil Gupta, Waqar Hasan, Hemant and Sonal Kanakia, Jitendra Malik, Purnima Mankekar, Arif Merchant, Inderpal Singh Mumick, Pandu Nayak, Shaibal Roy, Sriram and Uma Sankar, Ashok Subramanian, and Arun Swami.

- A special thanks to my American host family – Mona and David Blackburn for their love and encouragement.
- My brothers Kumar and Venkat
- And last but not least, my deepest debt is to my parents: to my father who taught me to dream, and to my mother who taught me to temper them with realism. It is to them that I affectionately dedicate this thesis.

Contents

List of Tables

List of Figures

Affectionately dedicated to
Appa and Amma.

Chapter 1

Introduction

1.1 The Need for Reformulation

One of the most important hypotheses in the field of artificial intelligence (AI) is the Physical Symbol System Hypothesis [NS76]: *Computation over symbolic representations is both necessary and sufficient for obtaining intelligent behaviour.* Over the last 30 years, research in AI in this paradigm has concentrated on developing sophisticated methods of controlling search in the space defined by a *fixed* representation provided by a human programmer. In this thesis, I investigate the complementary question of how to reduce or avoid search by changing representations automatically. This is the reformulation question first introduced in [New65].

The main motivation for reformulation stems from the observation that intelligent control of search cannot always fix problems caused by a bad representation. A dramatic example that illustrates this is the mutilated checkerboard problem [McC64]. Suppose we cut off two diagonally opposite corners of an 8 by 8 checkerboard. Can we cover this mutilated board by tiles of dimension 1 by 2? The naive formulation of this problem demonstrates the impossibility of achieving such a covering by trying *all* possible arrangements. A reconceptualization of this problem uses the fact that the two diagonally opposite squares are of the same colour and that each tile covers one square of one colour and one of the other colour. Since there are 30 squares of one colour and 32 of the other, there is no possible tiling. This reduces the solution that uses exhaustive search to a simple counting argument. How can a system discover this formulation of the problem? Newell and Simon pose this question in their Turing Award Lecture in 1975. They go on to add that:

> *The whole process of moving from one representation to another, and of discovering and evaluating representations, is largely unexplored territory in the domain of problem-solving research. The laws of quantitative structure governing representations remain to be discovered.*

The challenge above remains open even today: the power of most AI systems still lies in the representations given to them by their human designers. A first step to meeting this challenge is the theoretical investigation of, and the development of, tools to incrementally *re*-design representations—this is the reformulation problem. Automating incremental reformulations is very important, because it will go far towards relieving humans of the tedium and inflexibility of programming all possible conceptualizations of the world into AI systems.

The theoretical framework developed in the thesis is designed to automate the process of redesigning representations, which until now has been done entirely by humans. Our practical interest is in the design of a completely autonomous robot that functions well under resource constraints in changing environments. Intelligent agents have to survive in the world and also improve their strategic position with respect to it. They need to adapt to pressures from the environment by reformulating their view of the world to achieve their goals. Imagine a robot with a very detailed theory of the world. If the environment demanded faster prediction, the robot should build an approximate theory of the world on top of the more detailed one to allow for efficient computation. Both theories are about the same phenomena; the abstract theory is a reformulation that makes fewer distinctions and is thus much more efficient. Conversely, a system with a very crude theory, say about substances, would do very well to reformulate by introducing distinctions based on states of matter. A robot designed to pick objects off an assembly line should reformulate its conceptual hierarchy of objects to correspond to distinctions made on the basis of graspability.

In the mutilated checkerboard problem, replacing the initial set of distinctions based on board positions by the more abstract one based on colour (or board positions modulo 2) made the solution of the problem transparent. Reformulation is the science of removing irrelevant distinctions and introducing necessary ones to accomplish goals effectively and efficiently. Our aim in this thesis is to uncover general principles of change of conceptualizations which are notation- as well as domain-independent. We propose a general methodology for automating reformulations centered around these principles and instantiate it in the context of the specific problem of automating abstraction reformulations for computational efficiency.

But first, are there such general principles? And second, are they generative—that is, will they suggest new reformulations? If one sets out to propose a theory of reformulation that could generate all the spectacular reformulations in the history of science, it would appear that the answer, at least to the latter question, is negative. The reformulation of the geocentric model of planetary motion to the heliocentric one, as well as the frequency domain characterization of time-dependent behaviour by the Laplace transform, are special-case reconceptualizations that depended on then-undiscovered properties of problems in those domains. However, these revolutionary transformations account for a very small percentage of the reformulation phenomena in humans. Most reformulations are *incremental* and by comparison with the history of science examples, mundane. For instance, we conceptualize a road as a line when planning a trip, as a surface when we cross it, and as a volume when we attempt to dig it[1]. We move between interval- and instant-based representations of time fairly easily. These granularity [Hob85] transformations are fairly routine in humans. They are characterized by the incremental refinement or coarsening of distinctions made in the description of phenomena in order to make the goals of the agent easy to achieve. This thesis attempts to find invariants in granularity shifts and formulate them precisely enough to generate the shifts automatically.

1.2 Towards a Formulation of Reformulation

One of the main difficulties in posing the reformulation question comes from the fact that reformulation is an ill-understood phenomenon. Also, as with creativity, there is a mysticism associated with the ability to reformulate. Our first task is to isolate interesting and useful parts of the phenomenon that permit a relatively intuitive knowledge-based solution.

When we conceptualize a problem, we first identify the objects, functions and relations that are needed to state it. These elements represent the distinctions necessary to describe the domain as well as the goals. Reformulation is about changing distinctions:

[1] This example is thanks to Jerry Hobbs [Hob85].

changing the objects, functions and relations needed to formulate the problem. Semantically, reformulation is a shift in conceptualization. Conceptualizations are communicated to a machine by writing sentences in an appropriate language. Reformulation is achieved by changing representations or encodings of conceptualizations.

A reformulation is *correct* with respect to a set of goals or problems if the answers to the problems are preserved across the conceptual shift. The answer to the goal of tiling the mutilated checkerboard is preserved in the new formulation; the reconceptualization in this case is a correct one. Other standard examples of correct reformulations include the rectangular to polar coordinate transformations, and the Laplace and Fourier transforms. In all these cases, the basic primitives used to describe the problem are changed as a result of the reformulation. This shift in conceptualization causes a reconfiguration of the search space of solutions to a problem. For example, in the initial formulation of the mutilated checkerboard problem, there were 62 distinct objects; the reformulation grouped them into 30 objects of one colour and 32 of another. This regrouping causes the search space to shrink from size 2^{62} to 1. Reformulations are thus for a purpose: all of the conceptual transformations above lead to formulations that are computationally effective for certain classes of queries. A reformulation is called *good* if it leads to faster solution of the goals.

Before defining reconceptualizations, we provide a formal definition of a conceptualization.

Definition 1 *A conceptualization is a triple* $(\mathcal{O}, \mathcal{F}, \mathcal{R})$ *where* $\mathcal{O}$ *is a set of objects called the universe of discourse;* $\mathcal{F}$*, called the functional basis set, is a subset of functions from* $\mathcal{O}^n$ *to* $\mathcal{O}$*, and* $\mathcal{R}$*, called the relational basis set, is a subset of relations on* $\mathcal{R}^m$*, for n,m in the set of natural numbers.*

We can conceptualize kinship among a set of individuals as in Figure 1.1. A conceptualization makes our ontological commitments explicit. Another conceptualization of the kinship problem is in Figure 1.2. The *canonical language* $\mathcal{L}_\mathcal{C}$ of a conceptualization $\mathcal{C}$ has a distinct symbol name for every object, function and relation in $\mathcal{C}$.

Definition 2 *An encoding* $\mathcal{E}$ *of a conceptualization* $\mathcal{C}$ *is a set of sentences in the canonical language* $\mathcal{L}_\mathcal{C}$ *such that* $\mathcal{C}$ *is one of the models of* $\mathcal{E}$ *under Tarskian interpretation.*

An encoding of the conceptualization in Figure 1.1 is in Figure 1.3.

Objects: the set **P** of people {**A,B,C,D,E,F,G**}

Functions: the function **Father** from **P** to **P**, which is the set
{**(A,B),(A,C),(B,D),(B,E),(C,F),(C,G)**}

Relations: the relation **Ancestor**, which is the following subset of $\mathbf{P}^2$,
{**(A,B),(A,C),(A,D),(A,E),(A,F),(A,G),(B,D),(B,E),(C,F),(C,G)**}

the relation **SameFamily**, which is the set $\mathbf{P}^2$

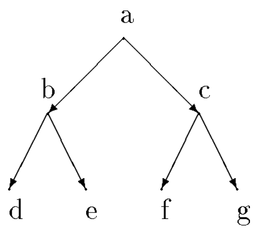
Figure 1.1: The conceptualization $\mathcal{C}_1$

Objects: the set **P** of people {**A,B,C,D,E,F,G**}

Relations: the relation **FoundingFather**, which is the following subset of $\mathbf{P}^2$,
{**(A,B),(A,C),(A,D),(A,E),(A,F),(A,G)**}

the relation **SameFamily**, which is the set $\mathbf{P}^2$

Figure 1.2: Another conceptualization $\mathcal{C}_2$



Father(a,b)
Father(a,c)
Father(b,d)
Father(b,e)
Father(c,f)
Father(c,g)
Father(x,y) $\Rightarrow$ Ancestor(x,y)
Ancestor(x,z) $\wedge$ Ancestor(z,y) $\Rightarrow$ Ancestor(x,y)

Figure 1.3: The encoding E_1 of $\mathcal{C}_1$

We will say that a conceptualization is *definable* in terms of another if it can be constructed from the other, along with some background knowledge in the form of another conceptualization Δ.

Definition 3 *A conceptualization $\mathcal{C}_2$ is a reconceptualization of $\mathcal{C}_1$ with respect to another conceptualization Δ if the elements of $\mathcal{C}_2$ are definable from $\mathcal{C}_1$ and Δ. The definitions of the elements of $\mathcal{C}_2$ in terms of Δ and $\mathcal{C}_1$ constitute the* articulation theory *between the two conceptualizations.*

In our kinship example, the conceptualization $\mathcal{C}_2$ can be constructed out of $\mathcal{C}_1$ by dropping the **Father** and **Ancestor** relations and introducing the new relation **FoundingFather** which is a subset of the *Ancestor* relation. Since $\Delta = \emptyset$ here, we have not introduced any new distinctions in this shift; we simply coarsened the **Ancestor** distinction. This reconceptualization is an *abstraction*. In the mutilated checkerboard problem, the new object—set of red squares—is definable in terms of the old conceptualization and set theory introduced via Δ. When Δ is non-empty, the new conceptualization makes distinctions not present in $\mathcal{C}_1$; we shall call such reconceptualizations *refinements*.

Definition 4 *$\mathcal{C}_2$ is a* correct *reconceptualization of $\mathcal{C}_1$ with respect to Δ and the set of goal relations G if G is definable in $\mathcal{C}_2$ only if it is definable in $\mathcal{C}_1$.*

A goal relation is definable in a conceptualization when it can be constructed in that conceptualization. In a correct reformulation, G is preserved exactly across the conceptual shift; G is also already definable in the initial conceptualization. These reformulations are called *deductive*. The abstraction of the conceptualization in Figure 1.1 to Figure 1.2 preserves the **SameFamily** relation; it is an instance of a deductive abstraction reformulation. A reformulation that makes an originally undefinable goal definable, as in the introduction of the concept *odd-integer* in the Lex system [Utg86], is an *inductive reformulation*.

Definition 5 *A set of sentences E_2 in the language L is a re-encoding of E_1 in the same language if the two sets of sentences have the same models.*

Whereas a description of reformulation at the level of conceptualizations captures correctness constraints, it is too coarse to model computation. To describe computational constraints on the solution of the goal, we use an encoding of the conceptualization and describe its computational properties with respect to a given problem solver. Since our chief interest in this thesis is in describing and automating reformulations for computational efficiency, we will define what it means for a reformulation to satisfy computational constraints.

Definition 6 *A reformulation $\mathcal{C}_2$ of the conceptualization $\mathcal{C}_1$ is* good *with respect to a problem solver PS and time and space bounds S on the computation of the goal wff g in $\mathcal{L}_{\mathcal{C}_2}$ if there is an encoding $\mathcal{E}$ of $\mathcal{C}_2$ that allows computation of g using PS within S. The interpretation of g in $\mathcal{C}_2$ is the goal relation G.*

The reformulation problem can now be described as follows:

Given
- the initial encoding $\mathcal{E}_1$ of the conceptualization $\mathcal{C}_1$
- A description of the problem solver PS.
- Correctness constraints: specification of the goal relation G
- Goodness constraints: time and space bounds on the computation of the goal relation.

Find
- a correct and good reconceptualization $\mathcal{C}_2$.

Before we proceed with a positive characterization of the space of reformulations for computational efficiency, it is worthwhile to recount that

Theorem 1 *There exist problems whose computational efficiency cannot be improved by reformulation.*

Proof: The Traveling Salesman Problem cannot be made easier to solve by a representation shift. The only way to improve efficiency is to change the solution criterion - i.e., accept a satisficing [Sim82] solution as opposed to the optimal one. □.

The study of NP-complete problems in theoretical computer science tells us that there are intrinsically hard problems— no clever representation or control strategy can reduce the complexity of such problems. The significance of NP-completeness results to reformulation is analogous to the significance of the second law of thermodynamics for physics and engineering; they tell us what is reasonable to attempt.

We represent formulations as partial logical theories and regard conceptualizations as their intended models. We modify the notion of a standard first order model to include not only the objects $\mathcal{O}$, but also the functions $\mathcal{F}$, and the relations $\mathcal{R}$. A model is thus the structure $(\mathcal{O}, \mathcal{F}, \mathcal{R})$. In this thesis we examine those shifts in formulation that correspond to reconceptualizations of their models, and which lead to faster solution of a given goal schema. In the traditional AI manner, we ask whether we have in-principle testers and generators for partial theories and their intended models.

1. Generation: Given a model $(\mathcal{O}, \mathcal{F}, \mathcal{R})$, what is the space of possible reconceptualizations that preserve answers to the goal? Given a partial theory, what is the space of reencodings of this theory?

2. Recognition: Given two conceptualizations, can we determine whether or not the goal schema is definable in both? Can we determine whether or not two reencodings are equivalent modulo a goal?

In Chapter 2, we consider the generation question at length. To give a flavour of the style of analysis, let us consider the class of information-losing or abstraction reformulations for the present. The space of possible reconceptualizations in this case consists of all those formulations whose conceptual primitives are *definable* from the given conceptualization. For a finite conceptualization with n_o objects, the number of definable unary relations is $\mathcal{O}(2^{n_o})$, whereas the number of binary relations is $\mathcal{O}(2^{{n_o}^2}$.

With respect to the recognition question too, we have the following mixed bag of results.

Theorem 2 *For finite conceptualizations $\mathcal{C}_1$ and $\mathcal{C}_2$, and a finite goal relation G, the problem of determining $Definable(G, \mathcal{C}_1) \equiv Definable(G, \mathcal{C}_2)$ is decidable.*

Proof: Insert proof here $\Box$.

Theorem 3 *For two encodings E_1 and E_2, and a goal schema g, the problem of determining $E_1 \vdash g \equiv E_2 \vdash g$ is undecidable.*

Proof: In general, it is undecidable whether or not two encodings E_1 and E_2 are equivalent with respect to a goal schema g. The proof proceeds by reduction to the halting problem and is in [Shm86]. $\Box$.

The significance of the above negative result is that a highly powerful mechanism for inventing completely novel representations is unlikely to exist. Also as pointed out by Simon, in *Models of Discovery Processes* [?], such a mechanism would have poor psychological plausibility because it would predict far more novelty than what occurs.

The positive conclusions to be drawn are

1. Only abstraction reformulations can be automated at the present, because we have an in-principle generator for the space of such reformulations.

2. Constraints on recognition restrict the form of the new abstraction. In particular, we will constrain all definitions to be universally quantified Horn formulae.

1.3 An Example

To instantiate the components of this problem, we present an example: this is a reformulation that is familiar to computer scientists — the re-representation of an equivalence relation by a partition.

- *The initial encoding and conceptualization*:

  We take the kinship problem conceptualized in Figure 1.1 and its encoding in Figure 1.3. Notice that the relation Ancestor is defined to be the transitive closure of Father. The goal is to determine whether or not two people in P belong to the same family: two people belong to the same family if they have a common ancestor.

- *The Problem Solver:*

  The problem solver that works on this formulation of the problem is a depth first backward chainer (e.g., Prolog). The axiomatic specification of the problem solver is in Figure 1.4. The simple cost model of problem solving actions shown in Figure 1.5 is used to determine how expensive the process of proof construction is in terms of time and space. The specifics of the time and space cost functions are not very important; the methodology proposed here works for any well-defined cost model.

- *The correctness and goodness constraints*:

  The correctness constraint is the preservation of the SameFamily relation. The reformulation and the present formulation have to behave identically with respect to this goal-schema. The goodness constraints are:

$$\begin{aligned}
\text{Solve(x)} &\leftarrow \text{Lookup(x)} \\
\text{Solve(x)} &\leftarrow \text{Lookup(x}\Leftarrow \text{y)} \wedge \text{Solve(y)} \\
\text{Solve(x}\wedge\text{y)} &\leftarrow \text{Solve(x)} \wedge \text{Solve(y)} \\
\text{Solve(}\neg\text{x)} &\leftarrow \text{Thnot(x) (negation by failure)}
\end{aligned}$$

Figure 1.4: A Simple Depth-First Backward-Chainer

$$\begin{aligned}
\text{Proof-Height(x)} &= \text{If x in Formulation then } c_1 \\
\text{Proof-Height(x)} &= \text{If x}\Leftarrow\text{y in Formulation then } c_2 + \text{Proof-Height(y)} \\
\text{Proof-Height(x}\wedge\text{y)} &= \text{Max(Proof-Height(x),Proof-Height(y))} \\
\text{Cost(Lookup(Ground Literal))} &= c_1 \\
\text{Cost(Lookup(x}\Leftarrow\text{y))} &= c_2
\end{aligned}$$

Figure 1.5: A Cost Model for the Problem Solver

- The new formulation should be able to solve any SameFamily query faster in the new formulation. In particular, we want these queries to be answered in O(1) time. Note that in the old formulations, this requires time proportional to the height of the Father tree.
- The new formulation can only consume as much space as the old formulation.

The answers that we expect are

- *The new conceptualization;*

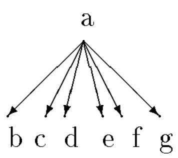
  The reconceptualization shown in Figure 1.2 satisfies the correctness constraints. The articulation theory is:

  1. The objects map over one to one.
  2. The new relation FoundingFather is described intensionally by the following definition:

$$\forall xy.\ \mathsf{FoundingFather}(x,y) \equiv \mathsf{Ancestor}(x,y) \wedge \neg\exists z.\mathsf{Ancestor}(z,x)$$

- *The new encoding:*

  The encoding shown in Figure 1.6 meets the goodness constraints because the computation of SameFamily can be achieved in constant time using two lookups on the



FoundingFather(a,b)
FoundingFather(a,c)
FoundingFather(a,d)
FoundingFather(a,e)
FoundingFather(a,f)
FoundingFather(a,g)
FoundingFather(z,x) $\wedge$ FoundingFather(z,y) $\Longrightarrow$ SameFamily(x,y)

Figure 1.6: The encoding E_2 of $\mathcal{C}_2$

> FoundingFather relation. Also, adding this new relation did not violate the space requirement.

This is an abstraction reformulation ($\Delta = \emptyset$). It is also *iso-ontic* because it did not change the objects. It is deductive because it preserves the goal relation and causes it to be computed faster.

1.4 Why Is Reformulation Hard?

The main stumbling block to automated reformulation is the fact that the intended interpretations of terms and symbols is never represented in the system itself. So a system has no logical basis for changing conceptualizations. A human would find it impossible to reformulate a description given entirely in terms of *gensysms*. Unfortunately, our reformulator is in no better position than the human with an uninterpreted description. The knowledge that is essential for reformulation consists of what the referents of the symbols are (i.e., the mapping between the ontology or conceptualization and the symbolic encoding), what *role* each element of the conceptualization plays in the computation of the goal, as well as the space of possible ways of perturbing the conceptualization. The reasoning needed to accomplish reformulations consists of means of evaluating the epistemic and computational consequences of perturbing the conceptualization, and designing encodings that correspond to a given conceptualization.

To do the above, we need a theory of representation as well as a theory of problem solving. A theory of reformulation can be seen as a bridge between these two theories. This viewpoint on reformulation gives us another insight into the difficulties of automating it. We have no theory of representation, and a fairly weak theory of problem solving [LJP87, TF85], so the construction of a strong theory of automating reformulations is impossible at this time. This thesis develops a theory of representation centered around the idea of *definability* of conceptual primitives and uses existing theories of problem solving to develop a general-purpose weak method for generating abstraction reformulations for computational efficiency.

1.5 Reformulations from First Principles

Previous attempts at reformulation have articulated the required knowledge in specific domains in highly compiled forms. The most representative of this class of research is Mostow's PhD thesis [Mos81] on the game of Hearts. This thesis attempts to explicate the origin of the compiled reformulation rules by a first-principles analysis that

- pins down the interpretation of the elements in a conceptualization by providing properties of the individual elements and constraints between them.
- makes the relationships between conceptualizations and encodings an explicit object to reason with.
- uses abstract representations of the conceptualization called *definability structures*, which allow analysis of the role of a conceptual element in the achievement of a goal.
- uses the properties of the problem solver and abstract representations of the proof and search spaces generated on particular encodings of the problem to reconfigure search spaces to meet computational constraints.

Our approach is to provide a unifying framework and a set of concepts that allow *declarative specification* of the knowledge that is required for automatic reformulation. Much of the knowledge about choice of conceptualization is left implicit, and that is why present day systems cannot change their conceptualizations in a justified way. For instance, the system cannot determine whether or not a vocabulary item makes the distinction it was designed for, especially in the face of changing environments. When computational constraints are changed, and the system has to realign boundaries, the presence of knowledge about the role of each conceptual element in the computation of the goal, makes it possible to evaluate *why* the present conceptualization fails to meet the constraints and determines *how* to fix it so as to achieve them.

The *justification-based* approach to reformulation makes the knowledge necessary for reformulation explicit and available for the reformulator to reason with. This knowledge is articulated as an explanation for a reformulation. There are constraints on the nature of this explanation: we construct them so that they can be inverted to *generate* reformulations. An explanation for a reformulation has two parts: correctness and goodness proofs. A correctness proof depends only on our theory of representation and it guarantees that the new formulation preserves a given class of queries with respect to the old one. A goodness proof shows that the new formulation has better computational properties than the old one: it requires taking the problem solver and our theory of problem solving into account. Standard proofs of correctness and goodness are non-generative. We use meta-theoretical justifications that tie the change in formulation directly to a change in computational properties. One class of such explanations are *irrelevance explanations.* An irrelevance explanation proves that certain distinctions in the formulation are not logically necessary for the solution of a given class of questions.

The crux of this thesis is the transformation of these justifications into generative procedures for choosing new terms in order to improve system performance. This is done by meta-theoretic reduction inferences that modify the formulation so that the irrelevance claims are no longer true of the new formulation. This is guided by a local optimisation principle called the *irrelevance principle* whose informal statement is: minimizing distinctions with respect to a set of goals, minimizes computational effort in the solution of these goals.

The minimization of distinctions irrelevant to the goal requires introducing new terms that stand for macro-objects in the formulation space and macro-actions in the search space. The reduction inferences restructure the computation using extra-logical criteria (e.g., minimize redundant computation) that bring the properties of the problem solver to bear, and a new formulation is obtained by regressing the restructured computation through an axiomatized description of the problem solver.

1.6 Claims of the Thesis

The general insights about reformulation and the process of automating it are

1. Reformulation is reconceptualization; a change in the objects, functions and relations assumed by a formulation. This level of description of the phenomenon captures an important invariant in the shift.

2. Abstraction reformulations can be automatically generated by the irrelevance principle which advises discarding of distinctions irrelevant to the goals at hand.

3. The irrelevance principle has a computational justification: it leads to the minimization of unnecessary computation.

4. Meta-theoretic claims of irrelevance about a formulation are key to the automatic generation of abstraction reformulations.

5. Meta-theoretic reduction inferences automatically eliminate irrelevance in a formulation.

The specific contributions of the thesis are

1. The development of the calculus of irrelevance that allows stating, verifying and generating justifications for abstraction reformulations.

2. The formulation of the irrelevance principle and a logical analysis of the meta-theoretic reduction of a formulation by irrelevance claims.

3. The design of algorithms for reduction that are graph-theoretic compilations of the reduction process.

4. The design of abstract representations for representations, called *definability structures* and the *definedness graph*, and efficient algorithms that operate on them.

5. Methods for the complete automation of the class of abstractions called *elimination of intermediates*.

1.7 Perspectives

1. Knowledge Representation: This thesis provides a semantic account of efficient language by describing reformulations for computational efficiency at the level of conceptualizations. The method of irrelevance minimization gives a generative account of how computational pressures shape representations. A new structure for describing representations, called *definability structures*, is also introduced.

2. Problem Solving: We introduce a new kind of meta-theoretic reasoning called irrelevance reasoning that speeds up computation of given queries in a logical formulation of a problem. It makes use of information about the queries, and the problem-solver that works on the queries, to determine what aspects of the formulation can be abstracted to make the computation efficient.

3. Machine Learning: The origin of new terms for computational efficiency, or skill-acquisition style of learning, is one of the open problems in machine learning. This thesis proposes an analytical solution to the problem that uses knowledge about the problem and the problem solver, as well as the principle of irrelevance to automatically acquire new vocabulary to improve performance.

4. Knowledge Engineering: We introduce the notion of irrelevance that is essential for knowledge base designers. Many of the justifications for choice of conceptualization can be formulated in terms of it. By explicitly recording these justifications, the designer can give the system the ability to modify its conceptualization automatically in the face of changing environments.

1.8 Reading Guide

This dissertation presents the thesis that representations are not arbitrarily chosen, rather they are the result of principles of computational economy. The document is organized as follows. Chapter 2 analyzes the reformulation problem and proposes a methodology for automating it. It also presents the irrelevance principle that underlies the generation of abstraction reformulations. The theoretical apparatus necessary for applying this principle is developed in Chapter 3. Chapter 4 shows how this principle can explain the formation of abstraction reformulations. While Chapters 3 and 4 provide an espistemologically adequate solution to automating reformulation, Chapter 5 addresses the heuristic adequacy of our solution. It describes special cases of the theory that are mechanizable and that cover a large percentage of abstraction reformulations. It also presents some results of empirical tests that confirm our theory. Chapter 6 restates the main results of the thesis and contains a discussion of their significance in designing representations. We conclude with an evaluation of the strengths and weaknesses of our approach and a proposal for future research on other kinds of reformulation.

There are some notational conventions we use in this thesis. We will distinguish things from symbols that represent them. Thus, elements of encodings will be printed in sans-serif while the elements of conceptualizations will be in **bold face**. As an example, the **Ancestor** relation will be represented by the relation symbol Ancestor. For propositional letters, we will use lower-case letters. For first-order formulas, we will distinguish between constants and variables. Constants will be in lower-case, variables in CAPITAL letters. The standard connectives $(\vee, \wedge, \neg, \Rightarrow, \equiv)$ and the quantifiers $(\forall, \exists)$ will be used.

Chapter 2

The Reformulation Problem

> *What can we do when we can't solve a problem? We can try to find a new way to look at it, to describe it in different terms. Reformulation is the most powerful way to attempt to escape from what seems to be a hopeless situation.*
>
> — Marvin Minsky, Society of Mind, page 141.

2.1 Introduction

Scientific advances involve not only solving problems but posing them as well. Asking the right questions is a creative act and solving them is a relatively routine activity, once the questions are correctly identified. This chapter attempts to describe the reformulation phenomenon as precisely as possible and poses the problem of automating it. Previous approaches to the problem are critically examined and the solution methodology proposed in this thesis is presented.

A cognitive phenomenon like reformulation can be explained at three different levels [Pyl84]: the intensional, semantics or knowledge level; the symbolic, syntactic or functional level; and the physical or biological level. An explanation of a phenomenon at the intensional level appeals to the semantic content of representations in an agent. The regularities of the phenomenon are captured by principles that mention the agent's goals and beliefs. In this thesis, we give a semantic account of reformulation as changing distinctions an agent makes in the world, in order to achieve its goals effectively. The symbolic account explains how these goals and beliefs are encoded and presents the algorithms by which the behaviour is achieved. Reformulation is realized by changing encodings: the symbolic account of reformulation is thus a syntactic one of theory change. The physical account explains how the symbol systems are actually constructed. A physical account of reformulation is not provided in this thesis, it would be useful for building a reformulation device.

2.2 The Semantics of Reformulation: Conceptual Change

Reformulation is reconceptualization. When one conceptualizes a problem, one determines what objects, functions and relations are needed for the purpose. The items in a conceptualization represent the distinctions for stating and solving the problem. Reformulation changes distinctions: it restructures our knowledge of the world in terms of new conceptual elements.

Traditional accounts of reformulation[Kor80, Mos81, Mar76a, New65, Len82] have only provided syntactic methods without the accompanying semantics. This thesis gives meaning to shifts in formulation by examining the shift in conceptualization it entails. By equating reformulation with reconceptualization we provide a clean Type-1 [Mar76b] theory of reformulation; i.e., we separate an account of what reformulation is, from how to do it. Before we describe what reconceptualizations are, we begin with some intuitions about, and a formalization of, conceptualizations.

2.2.1 Conceptualizations

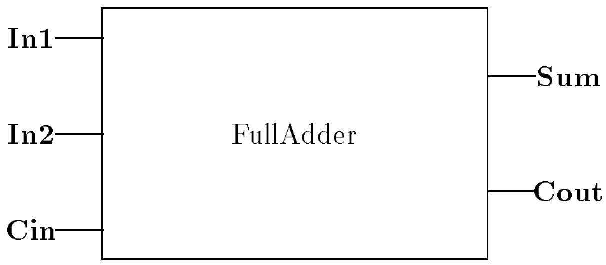
A conceptualization[GN87] is a model of the world. It consists of the objects, functions and relations that are of interest. For example, we can conceptualize kinship among a given set of individuals as in Figure 1.1. Another conceptualization of the kinship problem that preserves the **SameFamily** relation is shown in Figure 1.2. In the new conceptualization, the distinction between **Father** and **Ancestor** is removed and replaced by the maximal ancestor or the **FoundingFather** relation. A full adder can be conceptualized either at the gate level as in Figure 2.1 or as a unit, as in Figure 2.2. More examples of conceptualizations are found in [GN87]. Note that the items in bold face are relations in the world. This is our notational convention for the meta-language whose domain of discourse is the elements in a conceptualization.

We repeat the definition of a conceptualization from Chapter 1 for convenience.

Definition 7 *A conceptualization is a triple $(\mathcal{O}, \mathcal{F}, \mathcal{R})$ where $\mathcal{O}$ is a set of objects called*

Objects:	the gates: **X1**, **X2**, **A1**, **A2**, **O1**
	the ports: **a**, **b**, **c**, **d**, **e**, **f**, **sum**, **carry**
	the values: **0**, **1**
Functions:	
Relations:	**conn**,**and**, **or**, **xor**, **value**.

Figure 2.1: A conceptualization of a Full Adder



Objects:	the ports: **In1**, **In2**, **Cin**, **Cout**, **Sum**
Functions:	
Relations:	**and**, **or**, **xor**, **value**

Figure 2.2: Another conceptualization of a Full Adder

the universe of discourse; $\mathcal{F}$, called the functional basis set, is a subset of functions from $\mathcal{O}^n$ to $\mathcal{O}$, and $\mathcal{R}$, called the relational basis set, is the subset of relations on $\mathcal{R}^m$, for n,m in the set of natural numbers.

A conceptualization is a structure very similar to a Herbrand model – the only difference is that a Herbrand model consists of *all* the functions and relations defined on the Herbrand universe ($\mathcal{O}$), whereas we want to select particular functions and relations to be members of a conceptualization.

There is an interesting relationship between the elements in the two conceptualizations of the kinship problem. We can construct the relation **FoundingFather** out of the **Ancestor** relation by using standard relational operations [Ull82]. However, we cannot reconstruct the **Ancestor** relation out of the **FoundingFather** relation. The constructibility of a conceptual element from a set of such elements can be made precise using the logical notion of *definability* [End66].

Definition 8 *A conceptual element c is definable in terms of a set $\mathcal{C}$ of objects, functions and relations $x_1, x_2, \ldots, x_n$, written as Definable(c,C) if there exists a first-order formula ϕ with non-logical symbols γ, $\sigma_1, \sigma_2, \ldots, \sigma_n$, for which 1) there is a model of ϕ that maps each σ_i to the corresponding x_i and γ to c, and 2) Every model of ϕ that maps each σ_i to the corresponding x_i also maps γ to c. ϕ is called the defining formula for c in $\{x_1, x_2, \ldots x_n\}$.*

Notice that Definition 8 expresses the construction of c in terms of the elements of $\mathcal{C}$ *intensionally* as a formula in a language. Here are a few examples of the use of this definition.

Figure 2.3 shows a conceptualization of the blocks world. It consists of the blocks shown, and the relation **Clear** whose extension is the set of blocks that have no blocks on top of them,as well as the **On** relation that consists of block pairs (**x,y**) where **x** is on top of **y**. **On** = {(**A**,**B**), (**B**,**C**), (**E**,**F**)} and **Clear**={**A**,**E**}. The unary relation **Clear** is definable in terms of the binary **On** relation. The defining formula is

$$\mathsf{Clear}(x) \equiv \neg\exists y.\ \mathsf{On}(y,x)$$

The symbol On is interpreted to be the **On** relation. Every model of ϕ that maps On to **On** will have to map Clear to **Clear**. Note however, that **On** cannot be defined in terms of

Figure 2.3: Defining Clear in terms of On

Clear. This is because for a fixed model for Clear, the sentence $\neg\exists y.\ \mathsf{On}(y, x)$ constrains the set of possible models for On, but does not uniquely determine it.
In our kinship example, the **FoundingFather** relation is definable in terms of **Ancestor**, because we can construct a *definition*

$$\forall xy.\ \mathsf{FoundingFather}(x, y) \equiv \mathsf{Ancestor}(x, y) \wedge \neg\exists z.\ \mathsf{Ancestor}(z, x)$$

In all models that $\mathsf{Ancestor}$ refers to the **Ancestor** relation, the symbol $\mathsf{FoundingFather}$ is mapped to the **FoundingFather** relation.
Both the examples above are instances of defining a new relation in terms of other given relations. We now address the issue of defining new objects. One way to define a new object is to reify an existing relation. For instance, we can define the object **red** to stand for the predicate **Red** by introducing a new function from predicates to objects called the *denotation function* [McC79] and write

$$denotes(\mathbf{red}, \mathbf{Red})$$

. This would constitute the defining formula for the object **red** in terms of the relation **Red**.

Definition 8 uses *first-order definability*, because the formula ϕ is a first-order well-formed formula. In order to define the **SetofMissionaries** relation from a conceptualization that contains individual missionaries in the **Missionaries** relation, we need to be able to define arbitrary subsets of **Missionaries**. The defining formula ϕ is then

$$\mathsf{SetofMissionaries} = \{x \mid \mathsf{Missionary}(x)\}$$

. To see why this is not a pure first-order definition in the sense required by Definition 8, notice that the above definition can be rewritten as

$$\mathsf{member}(x, \mathsf{SetofMissionaries}) \equiv \mathsf{Missionary}(x)$$

There is no set of first-order axioms about member that makes standard set-theoretic reasoning reproducible inside first-order logic. We need to enrich our notion of a defining formula in Definition 8 to allow set-theoretic constructions. Then we can construct a richer class of conceptual elements from a given set.

Here's another example of a concept that is undefinable in a pure first-order system. The relation **Ancestor** is the transitive closure of the **Father** relation. Unfortunately, there is no first-order well-formed formula that defines the **Ancestor** relation in terms of the **Father** relation. The following

$$\forall xy.\mathsf{Ancestor}(x, y) \equiv \mathsf{Father}(x, y) \vee (\exists z.\ \mathsf{Ancestor}(x, z) \wedge \mathsf{Ancestor}(z, y))$$

constrains the **Ancestor** relation to be at least the transitive closure of **Father**, but does not fix it to be just the transitive closure. If this definition were interpreted within the semantics of Prolog (minimal Herbrand models), then the above sentence would constitute a valid definition for **Ancestor** under the conditions of Definition 8.
We can extend Definition 8 to cover the definability of an entire conceptualization in terms of another.

Definition 9 *A conceptualization $\mathcal{C}_2$ is definable in terms of a conceptualization $\mathcal{C}_1$, written as* Definable-C$(\mathcal{C}_1, \mathcal{C}_2)$, *if* $\forall c \in \mathcal{C}_2.\ Definable(c, \mathcal{C}_1)$. *The set of defining formulas constitutes the articulation theory between the two conceptualizations.*

The conceptualization in Figure 1.2 is definable from that in Figure 1.1. The articulation theory is the definition of the **FoundingFather** relation given before.

One of the problems with Definition 8 and thus with Definition 9 (since it hinges on Definition 8) is that it imposes almost no constraints on the nature of the defining formula ϕ. For finite conceptualizations, and a conceptual element with a finite extension, ϕ could simply be the trivial disjunction of the elements of the extension of that conceptual element. In the case of the kinship problem, we could as well have defined the **FoundingFather** relation directly as its extension over a certain universe of people. The problem with such a ϕ is that it needs to change when the universe of discourse changes.

Notice that the universally quantified defining formula provided before for the **FoundingFather** relation is independent of the particulars of the universe of discourse of the conceptualization.

To get around this problem, we define the notion of a *conceptual scheme* that abstracts away particular individuals in a conceptualization, and preserves the functional and relational structure. We can no longer describe such a conceptualization as in Figure 1.1, so we use an intensional description via definability claims. For the conceptualization in Fig 1.1, we have the following scheme:

$$Definable(\mathbf{Ancestor}, \{\mathbf{Father}, \mathbf{Ancestor}\})$$
$$Definable(\mathbf{SameFamily}, \{\mathbf{Ancestor}\})$$

Definition 10 *A conceptual scheme $\mathcal{CS}$ is a pair $(\mathcal{S}, \mathcal{D})$ where $\mathcal{S}$ is a set of relations and functions, and $\mathcal{D}$ is a set of definability claims of the form Definable(a,b) where $a \in S$ and $b \subseteq \mathcal{S}$.*

Since the identity of the defining formula ϕ is abstracted away in the relation *Definable*, the set of definability claims that constitute a conceptual scheme rarely pick out a unique conceptualization (i.e. they are not categorical, except in trivial cases). If on the other hand, we maintain ϕ along with the *Definable* relation, we essentially keep a recipe for constructing parts of a conceptualization from other parts. For instance **Father** is a base relation in the conceptualization $\mathcal{C}_1$. Given **Father** and the fact that *Definable-ϕ*(**Father**, **{Ancestor}**, Father(x,y) $\vee \exists z.$ Ancestor(x,z) $\wedge$ Ancestor(z,y)), we can construct the extension of the **Ancestor** relation. *Definable-ϕ* is important because it succinctly captures the construction of a conceptualization from its base objects, functions and relations. *Definable-C*, on the other hand, compactly describes the construction of one conceptualization from another. We define *Definable-C-ϕ* as the extension of *Definable-C* that maintains the articulation theory between the two conceptualizations.

2.2.2 Definability Analysis

Since definability is a key notion in our analysis of conceptualizations and reconceptualizations, we study its properties in detail.

Observation 1 Definable-C *is a reflexive relation.*

This follows directly from the definition of *Definable-C*. Every conceptualization can be constructed from itself by the identity map.

Observation 2 Definable-C *is a transitive relation.*

If we can construct $\mathcal{C}_2$ from $\mathcal{C}_1$ and the articulation theory is A_1, and if $\mathcal{C}_3$ can be constructed from $\mathcal{C}_2$ using the set of definitions A_2, then as long as the names of the elements in the three conceptualizations are standardized apart, we can construct $\mathcal{C}_3$ from $\mathcal{C}_1$ by composing the articulation theories in sequence.

Observation 3 Definable-C *is not symmetric.*

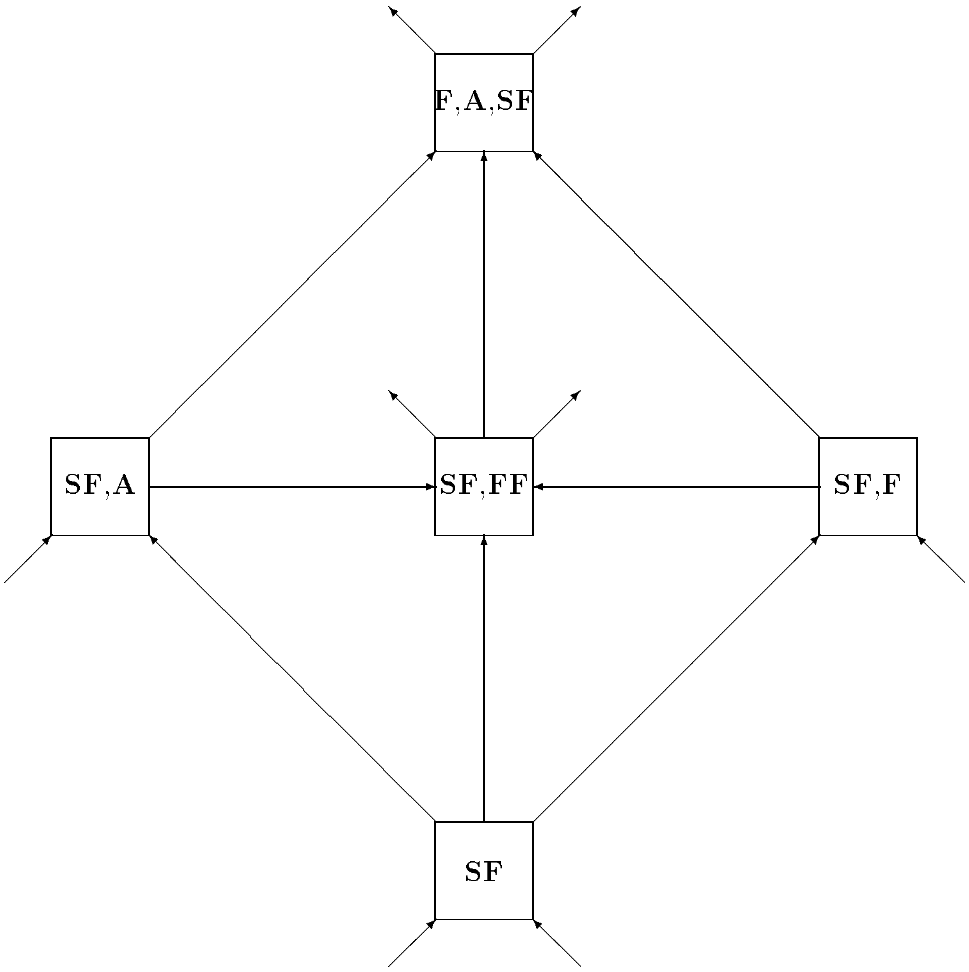
If $\mathcal{C}_2$ is created from $\mathcal{C}_1$ by losing information, then it is impossible to recover that information. This is the case in the abstraction of the conceptual primitive FoundingFather from the relation **Ancestor**. Unfortunately, *Definable-C* is not anti-symmetric either! If $\mathcal{C}_1$ is definable in terms of $\mathcal{C}_2$, and $\mathcal{C}_2$ is also definable in terms of $\mathcal{C}_1$, then $\mathcal{C}_1$ and $\mathcal{C}_2$ are only isomorphic and not identical. An example is the (r,θ) conceptualization and the (x,y) conceptualization of the real plane : each conceptualization can be constructed from the other, but they are not identical. Observation 3 prevents *Definable-C* from defining a lattice structure on conceptualizations.

Theorem 4 *Definable-C is a pre-order.*

Theorem 4 makes it easy for us to generate the search space of possible primitives for a conceptualization. Figure 2.2.2 introduces the *definability structure* which is a set of conceptualizations ordered by the *Definable-C* relation. From a given conceptualization $\mathcal{C}$ in a definability structure $\mathcal{S}$, we can construct the upper and lower sets of $\mathcal{C}$ both of which are subsets of $\mathcal{S}$.

$$
\begin{aligned}
Upper(\mathcal{C},\mathcal{S}) &= \{c \mid Definable\text{-}C(\mathcal{C},c) \wedge c \in \mathcal{S}\} \\
Lower(\mathcal{C},\mathcal{S}) &= \{c \mid Definable\text{-}C(c,\mathcal{C}) \wedge c \in \mathcal{S}\}
\end{aligned}
$$

The set *Upper* consists of conceptualizations that make finer distinctions than $\mathcal{C}$ and can thus be thought of as refinements of C. The set *Lower* contains conceptualizations that make coarser distinctions than $\mathcal{C}$, alternatively construed of as the abstractions of C. Both *Upper* and *Lower* are partially ordered by the relation *Definable-C*. Note that



Note: The arrows stand for the *Definable-C* relation, **F** for the **Father** relation, **A** for the **Ancestor** relation, and **SF** for the **SameFamily** relation.

Figure 2.4: A Definability Structure

$Upper(\mathcal{C},\mathcal{S}) \cap Lower(\mathcal{C},\mathcal{S}) = \mathcal{C} \cup \{c \mid Definable\text{-}C(c,\mathcal{C}) \wedge Definable\text{-}C(\mathcal{C},c) \wedge c \in \mathcal{S}\}$

We can define upper and lower bounds of two conceptualizations in $\mathcal{S}$ because of the existence of the pre-order *Definable-C*. A conceptualization $\mathcal{C}_l$ is a lower bound of the conceptualizations $\mathcal{C}_1$ and $\mathcal{C}_2$ if it is definable from both of them.

$Lower\_Bound(\mathcal{C}_1,\mathcal{C}_2) = \mathcal{C}_l \textit{ such that } Definable\text{-}C(\mathcal{C}_l,\mathcal{C}_1) \wedge Definable\text{-}C(\mathcal{C}_l,\mathcal{C}_2)$

Similarly, an upper bound of two elements in the pre-order is a conceptualization that defines them both.

$Upper\_Bound(\mathcal{C}_1,\mathcal{C}_2) = \mathcal{C}_u \textit{ such that } Definable\text{-}C(\mathcal{C}_1,\mathcal{C}_u) \wedge Definable\text{-}C(\mathcal{C}_2,\mathcal{C}_u)$

We can define greatest lower bounds and least upper bounds in the usual way. Informally, a lower bound of two conceptualizations includes at most the distinctions made in the intersection of the conceptualizations. An upper bound includes at least the distinctions made in the union of the conceptualizations. A greatest lower bound of two conceptualizations is an interesting structure, because it is one that preserves just the distinctions common to both conceptualizations. So the search for a minimal weakening of two conceptualizations that preserves some relations of interest, is simply the search for the greatest lower bound.

The definability structure is *bounded* because we only consider finite conceptualizations; the topmost node of the pre-order represents the finest grain of distinctions we can ever make in the world, and the lowest node is the empty conceptualization: it is definable in terms of every other node in the structure. We now present an example of how portions of this structure can be generated from a given conceptual scheme. We start with a schema consisting of the goal relation **SameFamily**, and we incrementally introduce finer distinctions. The generating formula for the first layer of nodes "above" this consist of two relation-schemas: **x** and **SameFamily** where *Definable*(**x**,{**SameFamily**}). The relations that satisfy this constraint are what we already know as **Father**, **Ancestor**, and **FoundingFather**. These **x**'s are called *interpolants* of **SameFamily**. We can build further nodes by constructing interpolants of the newly introduced relations. This is shown in Figure 2.4.

2.2.3 Reconceptualizations

Definability is an important tool for analyzing the relationship between different conceptualizations because it succintly describes how one conceptualization can be constructed

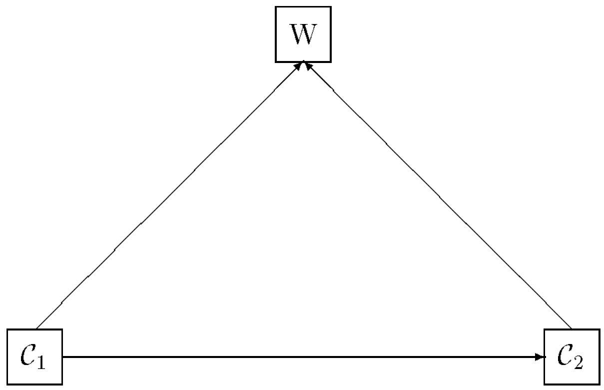


Figure 2.5: Conceptualizations and the World

from another. We can now define what we mean by a conceptual shift.[1]

Definition 11 *A conceptualization $\mathcal{C}_2$ is a reconceptualization of $\mathcal{C}_1$ with respect to some background conceptualization Δ, if the elements of $\mathcal{C}_2$ are definable from $\mathcal{C}_1$ and Δ.*

Reformulation is the science of introducing and removing distinctions. It is the ability to cut the world up into the right pieces for the current goals. The items in a conceptualization denote distinctions in the world. The reconceptualization is also *about* this world. How do we capture this notion? The denotation relation between the world and the conceptualization as shown in Figure 2.5 is in the head of the modeler. To get at the fact that both conceptualizations are about the same world, we rely on the integrity of the denotation relation between $\mathcal{C}_1$ and the world, and then construct $\mathcal{C}_2$ out of $\mathcal{C}_1$. This is why definability is an important constraint in our definition of reconceptualization. If Δ is the null set, then the new conceptualization is guaranteed to be about the same world[2], since definable relations and objects can only make existing distinctions coarser and cannot introduce new ones. Definability captures the notion of retiling a given $\mathcal{W}$ or an abstraction of it. If we wish to introduce distinctions, we enrich $\mathcal{W}$ explicitly through the background knowledge Δ. For instance, the introduction of the sets of missionaries and cannibals is done by the introduction of set theory through Δ.

[1] This is a restatement of Definition 3 of Chapter 1.

[2] or as much about the given world as the original conceptualization was.

We notice that according to Definition 5, the conceptualization $\mathcal{C}_2$ in Figure 1.2 is a reconceptualization of $\mathcal{C}_1$ in Figure 1.1, with $\Delta = \emptyset$. This reconceptualization did not change the objects in the domain of discourse, only the relations on them. It is called an *iso-ontic* reconceptualization. Since it is the case that

$$\neg\ \textit{Definable}(\textbf{Ancestor}, \{\textbf{FoundingFather}\}),$$

the reconceptualization has lost information; we call it an *abstraction*. An example of a reconceptualization that involves changing the objects is the classic missionaries and cannibals puzzle, first analyzed in [Ama68]. The initial conceptualization of the problem consists of six individuals: Larry, Curly and Moe (the missionaries), and Huey, Dewey and Louie (the cannibals). The reconceptualized problem introduces two new objects: the set of missionaries and the set of cannibals. The introduction of the sets allows us to change the granularity of the individual-based boat load and unload actions into set-based load and unload actions. This then allows for efficient computation of the schedule for transporting people across the river.

Let us study some more cases of conceptual change and see whether they fit our definition. The change from the geocentric into the heliocentric conceptualization of planetary motion was an iso-ontic reconceptualization[3] that changed the relations among the members of our solar system. Central to the geocentric theory was the relation **Orbits-Earth**, whose extension included the sun and all the planets except the earth. The heliocentric theory grouped the objects somewhat differently, it posited the existence of the **Orbits-Sun** relation whose extension included all the planets. Note that being a finite relation, **Orbits-Sun** is definable in terms of **Orbits-Earth** and equality.

$$\forall x.\ \mathsf{Orbits\text{-}Sun(x)} \equiv [x \neq sun \wedge \mathsf{Orbits\text{-}Earth(x)}\ ]\ \vee\ x = earth$$

The same phenomena (the astronomical data) was now explained in a simpler way in the heliocentric theory with the new conceptual primitive **Orbits-Sun**. Conceptual change (change in the primitives to describe the world) and theory revision (change in what we say about the world) often go hand-in-hand and it is difficult to separate the two phenomena. There are at least three ways in which different conceptualizations of a given world differ[Car87].

[3] we shall treat epicycles as relations and not objects, for this purpose.

1. *Individual concepts in the system*

   A pure case of conceptual change occurs in the Fourier and Laplace transformations as well as rectangular to polar coordinate transforms. Reformulating a description of a 2-D scene expressed in rectangular coordinates into polar coordinates, preserves the content of the intial description exactly. The distinctions made by the two conceptualizations are different, they tile the same content space in different ways.

2. *The domain of the phenomena accounted for*

   An example from physics is the wave and particle conceptualizations of light. Each conceptualization accounted for phenomena the other couldn't. A more mundane case is the difference in the conceptualization of a leaf by a botanist and by a layman. The botanist makes many more distinctions than the layman because he needs finer distinctions to be able to make predictions of interest to him. Yet another case from the history of science is the shift from the Aristotelian to the Galilean theory of motion: Aristotle's theory included all changes over time: growth, decay, movement, etc, whereas Galileo specialized it to cover movement alone [Kuh87].

3. *The nature of the explanation*

   The shift to the heliocentric theory made the explanations of astronomical observations simpler. This is a case where the nature of explanations changed as a result of the conceptual shift. In our kinship example, an encoding of the reconceptualization in terms of *FoundingFather* makes proofs of *SameFamily* propositions shorter. The novice-expert conceptual shifts studied in detail by cognitive scientists [CME82] indicate that experts use relations among objects that are definable in terms of those held by novices. However, possession of these concepts allow them to interpret a problem situation better and state strategies in a much more perspicuous fashion than is allowed by the novice's ontology.

It should be emphasized that identifying the ontological commitments of a theory (especially scientific ones) is a non-trivial endeavour. Our definition of a conceptual change allows us compare two *given* ontologies and assess whether they are *about* the same reality by determining their interdefinability.

We shall call a reconceptualization *correct* with respect to a set of goal relations, if the goals are preserved in the new conceptualization. More formally,

Definition 12 *$\mathcal{C}_2$ is a* correct *reconceptualization of $\mathcal{C}_1$ with respect to the set of goal relations* $\mathbf{G}$*, if Definable($\mathbf{G}$,$\mathcal{C}_1$) if and only if Definable($\mathbf{G}$,$\mathcal{C}_2$).*

The kinship reformulation is correct with respect to the goal relation **SameFamily** and incorrect with respect to the goal relation **Father**.

The search for a correct reconceptualization in the definability structure $\mathcal{S}$ is the search from the given $\mathcal{C}$ to one where the goal $\mathbf{G}$ is still definable. A principle of economy that dictates the making of the fewest distinctions would allow us to pick the lowest node in the definability graph that has $\mathbf{G}$ definable in it. The role of the background conceptualization Δ is to help navigate in the space of alternative conceptualizations $\mathcal{S}$, either by directed addition of distinctions in traversing $Upper(\mathcal{C}, \mathcal{S})$, or the directed losing of distinctions while traversing $Lower(\mathcal{C}, \mathcal{S})$.

We can distinguish several types of reconceptualizations

1. New conceptualization definable in terms of the old one

   This class includes abstraction reformulations like the kinship example and the full adder example. There are two basic types of operations needed to generate them from a given conceptualization: dropping conceptual elements (e.g., the removal of the relation **Father**), and adding definable compounds (e.g., the addition of the relation **FoundingFather**). This class of reconceptualizations does not permit the solution of goals that were unsolvable in the old conceptualization. Often they set the stage for encoding shifts that permit faster solution of previously soluble goals.

2. New conceptualization consistent with the old one

   The operation that generates conceptualizations in this class is the addition of new conceptual elements that are not definable in terms of the existing primitives. An example is the addition of epicycles to the Ptolemaic conception of planetary motion. This addition usually allows for the solution of goals that couldn't be solved before. The goals that could be solved in the old conceptualization remain solvable and yield the same answers in the reconceptualization.

3. New conceptualization inconsistent with old conceptualization

   An example is the shift from the Ptolemaic to the Galilean conceptualization. Some of the predictions made by the Ptolemaic theory were no longer made by the Galilean one. However, the two conceptualization had some common elements: the observed astronomical data, and the objects in the solar system.

2.2.4 A Knowledge Level Analysis of Reformulation

Until now, we have discussed what a reformulation is, and how to generate the space of possible reconceptualizations. We now turn our attention to the role reformulation plays in an intelligent agent, so as to be able to navigate the definability structure in a goal-directed manner. Why is reformulation an intelligent thing to do, and under what conditions should an agent reformulate? To answer these, we perform a knowledge level analysis of a reformulator. This requires us to in turn answer the following.

1. What goals do we ascribe to a reformulator?
2. What other beliefs do we ascribe to it?
3. What is the nature of the background knowledge that together with the rationality principle generates the appropriate reformulation behaviour?

Note that a reformulation action is somewhat different from a traditional action like *MoveBlock* that changes the current state of the world. A reformulation action does not the change the world, it causes the agent to redescribe the world in its head. Because of this redescription, an agent's actions in the world might be affected. An agent that was unable to achieve a goal in a previous conceptualization, might be able to achieve it in a reformulated version; this would be the secondary effect of a reformulation action. Clearly, all useful reformulations have interesting secondary effects.

The goal of a reformulator is to redescribe the world in terms of primitives (objects, functions and relations) that would permit the effective solution of a specified class of problem-solving goals. The beliefs that we attribute to the reformulator include

1. an initial conceptualization of the world.
2. the correctness constraints described by the problem-solving goals.
3. an initial encoding of that conceptualization together with its computational properties with respect to a given problem solver.
4. the effectiveness constraints imposed by the environment.

Whereas a conceptualization is fine-grained enough to capture correctness constraints, it is too coarse to model computation. To describe computational constraints on the solution of the goal, we need to introduce the concept of an *encoding* of a conceptualization.

An encoding is simply a set of sentences in an appropriate language, one of whose models is the conceptualization. Details of the relationship between the encoding and the conceptualization will be left till Section 3. For now, we will assume that there is a space of encodings associated with a conceptualization and that effectiveness constraints are with respect to computation of the goal relations in an encoding.

The goal of the agent is to solve its goals within the given correctness and effectiveness constraints. The impetus to reformulate arises out of the fact that the agent cannot meet these constraints, and thus decides to incrementally reconceptualize the world to achieve them. The *logical problem* of reformulation is: what other general beliefs do we assign to the reformulator that would allow it to deduce the new conceptualization?

To see what is needed, consider the deduction of the new primitive **FoundingFather** from the conceptualization $\mathcal{C}_1$ in Figure 1.1. The space of reconceptualizations for the goal relation *SameFamily* is in Figure 2.4. We will assume that the time and space constraints on the computation of the goal are met in conceptualizations in which the primitive *FoundingFather* occurs[4]. The minimality principle of making just enough distinctions to meet the correctness and goodness constraints dictates the choice of the conceptualization {**FoundingFather**, **SameFamily**}.

Thus, the knowledge that we attribute to the reformulator includes: knowledge of the space of reconceptualizations, knowledge to determine whether or not a particular conceptualization and a particular encoding of it, meet the correctness and effectiveness constraints respectively, as well as knowledge of a principle of economy in the choice of conceptualizations. This would logically entail a new choice of conceptual elements that solves for the goal within the given computational constraints.

2.3 A Syntactic Account of Reformulation: Theory Change

A conceptualization is an extensional description of the phenomenon of interest. It is typically in the head of the programmer; she communicates it to the agent by writing sentences in a language that is appropriate to that conceptualization. A *language* is a set of sentences with a specified syntax and semantics. The syntax of a language defines the sentences legal in that language. The semantics of a language defines the relationship

[4] The determination of whether an encoding satisfies some effectiveness constraint is discussed in Section 3.2.

Father(a, b)	Ancestor(a, b)
Father(a, c)	Ancestor(a, c)
Father(b, d)	Ancestor(b, d)
Father(b, e)	Ancestor(b, e)
Father(c, f)	Ancestor(c, f)
Father(c, g)	Ancestor(c, g)
	Ancestor(a, d)
	Ancestor(a, e)
	Ancestor(a, f)
	Ancestor(a, g)

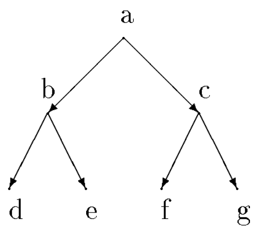
Figure 2.6: The Canonical Encoding for $\mathcal{C}_1$

between the sentences and the programmer's conceptualization of the world. This relationship is called an *interpretation*: it consists of a mapping between the symbols of the language and the objects, functions and relations in the conceptualization, as well as rules for determining the truth of sentences composed of these symbols.

First-order predicate calculus is the language used in this thesis. We could use a specialized language (e.g. that of trees or graphs, musical scores, flowcharts and electrical circuits) to encode our conceptualization. However, we take the position expressed in [Hay81] and expanded in [MG84] that specialized languages can be understood in terms of their translations into first-order theories. To encode a conceptualization in first-order predicate calculus, we need to select the non-logical symbols that denote the various elements of the conceptualization. One such choice is the canonical language introduced in Section 1.4.

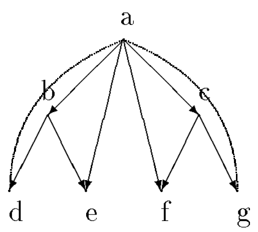
Definition 13 *The canonical encoding of a conceptualization $\mathcal{C}$ is in the canonical language $\mathcal{L}_{\mathcal{C}}$ and lists all the tuples of the functions and the relations.*

The interpretation function for canonical encodings is particularly straightforward: it is a 1-1 map. The canonical encoding for the kinship conceptualization $\mathcal{C}_1$ is in Figure 2.6. An example of a non-canonical encoding that uses a canonical language is in Figure 1.3. Yet another non-canonical encoding for $\mathcal{C}_1$ is in Figure 2.7. Note that the encodings E_3 and E_1 differ on the actual definition of the Ancestor relation. Both encodings however have the same model: the conceptualization $\mathcal{C}_1$. Encoding E_4 of $\mathcal{C}_1$ is displayed in Figure 2.8. E_4 differs from both E_1 and E_3 in its commitments to which relations are primitive and which are computed. Whereas, Father is explicitly recorded in E_1 and E_3 and Ancestor



Father(a,b)
Father(a,c)
Father(b,d)
Father(b,e)
Father(c,f)
Father(c,g)
Father(x,y) $\Rightarrow$ Ancestor(x,y)
Father(x,z) $\wedge$ Ancestor(z,y) $\Rightarrow$ Ancestor(x,y)

Figure 2.7: The encoding E_3 of $\mathcal{C}_1$



Ancestor(a,b)
Ancestor(a,c)
Ancestor(a,d)
Ancestor(a,e)
Ancestor(a,f)
Ancestor(b,d)
Ancestor(b,e)
Ancestor(c,f)
Ancestor(c,g)
Ancestor(a,g)
Ancestor(x,y) $\wedge\neg\exists z.$ Ancestor(x,z) $\Rightarrow$ Father(x,y)

Figure 2.8: The encoding E_4 of $\mathcal{C}_1$

is computed in terms of it, E_4 makes Ancestor the primitive relation and defines Father using the Ancestor relation.

The exact relationship between an encoding and a conceptualization can be formalized using the framework of first-order logic. We will view a conceptualization as a special kind of a structure (akin to a model, except that we reify functions and relations), and encodings as axiomatizations in first order logic. To relate a conceptualization $\mathcal{C}_1$ and an encoding E_3, we find an interpretation function I from the symbols in E_3 to the objects in $\mathcal{C}_1$ such that every sentence in the encoding holds in the conceptualization. For encoding E_3 the interpretation I is:

$I(\mathsf{a}) = \mathbf{A}$
$I(\mathsf{b}) = \mathbf{B}$
$I(\mathsf{c}) = \mathbf{C}$
$I(\mathsf{D}) = \mathbf{D}$
$I(\mathsf{e}) = \mathbf{E}$
$I(\mathsf{f}) = \mathbf{F}$
$I(\mathsf{g}) = \mathbf{G}$
$I(\mathsf{Father}) = \{(\mathbf{A},\mathbf{B}),(\mathbf{A},\mathbf{C}),(\mathbf{B},\mathbf{D}),(\mathbf{B},\mathbf{E}),(\mathbf{C},\mathbf{F}),(\mathbf{C},\mathbf{G})\}$
$I(\mathsf{Ancestor}) = \{(\mathbf{A},\mathbf{B}),(\mathbf{A},\mathbf{C}),(\mathbf{A},\mathbf{D}),(\mathbf{A},\mathbf{E}),(\mathbf{A},\mathbf{F}),(\mathbf{A},\mathbf{G}),(\mathbf{B},\mathbf{D}),(\mathbf{B},\mathbf{E}),(\mathbf{C},\mathbf{F}),(\mathbf{C},\mathbf{G})\}$

An encoding E models a conceptualization $\mathcal{C} = (\mathcal{O},\mathcal{F},\mathcal{R})$ exactly when $\mathcal{C}$ satisfies all the formulae in E in the sense defined below.
A well-formed formula $\phi \in E$ holds in a conceptualization $C = (\mathcal{O},\mathcal{F},\mathcal{R})$ if we can find an interpretation function I from the set of parameters to items in C such that

1. Every $\forall$ quantifier symbol is assigned the set $\mathcal{O}$.
2. Every constant symbol c is assigned an element c^I in $\mathcal{O}$.
3. Every n-place predicate symbol is assigned the corresponding n-place relation from $\mathcal{R}$.
4. Every n-place function symbol is assigned the corresponding n-place function in $\mathcal{F}$.

We define the function s_o that maps individual variables to elements in the set $\mathcal{O}$. Now we extend s_o to name all terms in $\mathcal{L}(C)$. The function that translates a term to the object it denotes is $\overline{s}_o$. It is defined recursively.

$$\begin{tikzcd}
\mathcal{C}_1 \arrow[r, "\mathcal{R}_{sem}"] & \mathcal{C}_2 \\
E_1 \arrow[u, "I_1"] \arrow[r, "\mathcal{R}_{syn}"'] & E_2 \arrow[u, "I_2"']
\end{tikzcd}$$

Figure 2.9: Reconceptualization and Reencoding

$\boxed{\text{For a } g \in E_1 \;\; \mathcal{C}_2 \models \mathcal{R}_{syn}(g) \;\text{ if and only if }\; \mathcal{R}_{sem}(\mathcal{C}_2) \models g}$

1. For each variable x, $\overline{s}_o(x) = s_o(x)$.

2. For each constant symbol c, $\overline{s}_o(c) = c^I$.

3. If $t_1, t_2, \ldots, t_n$ are terms, $\overline{s}_o(f(t_1, t_2, \ldots, t_n) = f^I(\overline{s}_o(t_1), \overline{s}_o(t_2), \ldots, \overline{s}_o(t_n))$.

Now that we have defined how terms denote objects, we can specify how the truth of formulas is determined. This is done by recursion on the structure of the well-formed formulas.

1. Base case: ϕ is an atomic formula and is of the form $P(t_1, t_2, \ldots t_n)$. We say that $\models_I \phi$ if and only if $(\overline{s}_o(t_1), \overline{s}_o(t_2), \ldots, \overline{s}_o(t_n)) \in P^I$.

2. Recursive case 1: ϕ is of the form $\neg\phi_1$. $\models_I \phi$ if and only if $\not\models_I \phi_1$.

3. Recursive case 2: ϕ is of the form $\phi_1 \wedge \phi_2$.
   $\models_I \phi$ if and only if $\models_I \phi_1$ and $\models_I \phi_2$.

Now that we understand the relationship between a conceptualization, and it encoding, we can state the connections between changes in conceptualization and changes in encoding more formally. Let $\mathcal{R}_{sem}$: $\mathcal{C}_2 \longrightarrow \mathcal{C}_1$ be the articulation theory between $\mathcal{C}_1$ and $\mathcal{C}_2$. Recall that this articulation theory contains the definitions of the elements in $\mathcal{C}_2$ in terms of those in $\mathcal{C}_1$. Let $\mathcal{R}_{syn}$: $E_1 \longrightarrow E_2$ be the mapping between the encodings E_1 of $\mathcal{C}_1$ and E_2 of $\mathcal{C}_2$. The two mappings $\mathcal{R}_{sem}$ and $\mathcal{R}_{syn}$ are related as shown in Figure 2.9.

We factor changes in primitives from changes in their encoding. We structure our search for a good reformulation as a search through the space of conceptualizations guided by the computational constraints checked in the much denser space of encodings. The reason that the encoding space is much denser is that not all shifts in encodings correspond to a shift in conceptualization. For instance, the shift in encoding from E_1 to E_3 or from E_3 to E_4 cause no change in conceptualization. The differences between these encodings surface in

1. Choice of which element to store and which to compute

2. The actual definition of an element in terms of the others

These are changes in the symbol level with no accompanying change at the knowledge level. These shifts do not qualify as reformulations within our framework.

For the three categories of conceptual shifts in Section 2.2.2, we provide encoding shifts that implement them.

1. New conceptualization definable in terms of the old one
   The shift in encoding that achieves this class is the introduction of new terms with the appropriate definitions and the re-axiomatization of the old encoding using these new (eliminable) defined terms. A good example of this is the introduction of the FoundingFather symbol in the encoding E_1 and the re-axiomatization of SameFamily in terms of it, to generate E_2.

2. New conceptualization consistent with the old one
   The encoding shift that accomplishes this is the introduction of new non-eliminable terms and subsequent re-axiomatization using these terms. The new encoding is a consistent extension of E_1.

3. New conceptualization inconsistent with old conceptualization
   Implementing this involves dropping and adding new terms as before and re-axiomatizations that make non-monotonic changes to the encoding.

2.4 A Catalogue of Examples

Reformulation is a diverse phenomenon as the following set of examples indicate. In all these cases, we note that there is a change in the basic terms used to describe a problem. A classic example is rewriting the missionaries and cannibals (M and C) problem [Ama68] phrased in terms of individuals into a formulation that is based on the cardinalities of the *sets* of missionaries and cannibals. This is a reformulation that improves the computational efficiency of solution of the M and C problem. Other examples are

1. The Copernican theory of planetary motion is a reformulation of the geocentric theory. Both theories assume the same objects and functions. However the relations among the objects assumed differ in the two systems. The conceptual shift preserves the observed data about the motion of the various planets. This reformulation is akin to a shift of origin in coordinate geometry.

2. Reformulation of a theory in terms of another. In his thesis on reformulation in 1976, Mark [Mar76a], reformulates the managerial problem of hiring in a firm in terms of Keynesian economics. Minsky argues [Min86] that this is the way humans understand new things: by casting them in terms of familiar theories. Most engineers map problems in heat conduction into corresponding problems in analog circuits since both these domains share the same behavioral abstractions [Gre85].

3. Granularity shifts

   (a) Temporal: Shifting between instant and interval representations of time is essential for building efficient planners that deal with time.

   (b) Spatial: A road is viewed as a line for the purposes of planning a trip, a surface when one crosses it and as a volume for digging it. Most reasoners about the common sense physical world need to be able to shift between these views of space when appropriate. Jerry Hobbs [Hob85] outlines a scheme whereby we can capture the connections between the views as logical theories.

   (c) Aggregation of objects: the full adder abstraction is an example from digital circuits. The notion of Thevenin and Norton equivalents in analog circuits is a reformulation that aggregates a large number of circuit elements into one lumped parameter.

(d) Equivalence class reformulations: partitioning the integers into odd and even as in the checkerboard reformulation and the introduction of "d" in propagation of faults in digital circuits are examples of this phenomenon.

4. The Aha! Insight reformulations. These include reformulations like the Laplace and Fourier transforms, the star-delta transformation vin analog circuits, as well as the polar-rectangular coordinate transformations in which there is no loss of information across the conceptual shift. These are impossible to automate.

5. Making symmetries explicit

6. Control reformulations: An example is the reformulation that transforms the naive formulation of the Fibonacci function which has exponential complexity to a tail-recursive one that is linear. These reformulations are called control reformulations because they can be equally well implemented by changing the control strategy of a problem solver while keeping the formulation intact.

7. Change of perspective: these involve changing what constitutes the "figure" and the "ground" elements in a conceptualization. A good example is the re-conceptualization of the 8 puzzle where the actions are initially expressed in terms of the tiles, into one where the actions are expressed in terms of the movements of the blank tile.

8. Data structure reformulations: These cover pure encoding transformations with no change in conceptualization. Ordering conjuncts in a query, storing a relation in a hash table instead of a list are examples.

9. Notational variants: a classic example is the reformulation of Roman numerals to Arabic. These are very hard to discover automatically, because the space of notational variations is hard to describe.

10. Structure-function reformulations: Minsky's arch example is an instance. If arches are described in structural terms (blocks, and the support relations between blocks), it would be difficult for a system to recognize structurally different arches as instances of the same basic concept. If however, arches were described in terms of the functionally motivated predicates *Body* and *Support* which stand for the top and bottom parts of an arch, recognition would be trivial.

11. Reformulations that cause compression of reasoning chains: these generally do not involve introducing new objects and include constant folding and loop jamming from compiler optimisations and computing prejoins in database query optimisations. Another example is compiling a rule set for diagnosing a circuit from a description of its structure and behaviour.

12. Reformulations from an objective to a subjective ontology of the world. Subjective ontologies have been used by Agre and Chapman [AC87] as well as Brooks [Bro87] to build agents that act in the world in real time. Subjective ontologies are shown to be reformulations of the objective ontology of the world assumed by the situation calculus [SW89].

2.5 Taxonomy of reformulations

Reformulations are of two basic kinds: inductive and deductive. Deductive reformulations lead to the formation of new conceptualizations where a class of goals solvable in the original formulation is solved *faster*. Inductive reformulations are those in which the new conceptualization solves goals that couldn't be solved before. The reconceptualizations in the missionaries and cannibals problem as well as the kinship reformulation are deductive. An example of an inductive reformulation is the structure-function example from above.

For deductive reformulations, we can specify correctness constraints, viz., the goals that need to be preserved across the conceptual shift, as well as the goodness constraints, viz., the bounds on time and space in computing the goal formulas in an encoding of the reconceptualization. In inductive reformulations, the clean separation between correctness and goodness does not obtain. Utility is the chief issue (i.e., what is good is correct). Inductive reformulations are described in [RS88]. We focus on the problem of automating deductive reformulations in this thesis.

Deductive reformulations themselves come in three categories: abstractions, refinements and isomorphisms. A reconceptualization $\mathcal{C}_2$ of $\mathcal{C}_1$ is

1. **An Abstraction**: if *Definable-C*($\mathcal{C}_2$,$\mathcal{C}_1$) and $\mathcal{C}_2$ is a correct reformulation of $\mathcal{C}_1$ according to Definition 12 with respect to some goals **G**. The kinship reformulation of Chapter 1 is an example.

2. **A Refinement**: if *Definable-C*($\mathcal{C}_1$,$\mathcal{C}_2$) and $\mathcal{C}_2$ is a correct reformulation of $\mathcal{C}_1$ according to Definition 12 with respect to some goals **G**. Consider the following formulation of a puzzle called the hermit puzzle. A hermit starts at the bottom of a hill at 8 am one morning and climbs to the top by 5 pm. He returns to the bottom of the hill the next day at 5 pm after starting his journey at the top at 8 am. The goal is to show that there is some point on the path up the hill that the hermit passed at the same time but on different days. Introducing another hermit who comes down the hill the same time that the first one starts makes the solution transparent. The new conceptualization introduced an object into the formulation that cannot be constructed from the old one: it is thus a refinement reformulation.

3. **An Isomorphism**: if *Definable-C*($\mathcal{C}_2$,$\mathcal{C}_1$) and if *Definable-C*($\mathcal{C}_1$,$\mathcal{C}_2$) and and $\mathcal{C}_2$ is a correct reformulation of $\mathcal{C}_1$ according to Definition 12 with respect to some goals **G**. The rectangular to polar coordinate transformation in coordinate geometry is an instance of an isomorphic reformulation.

This thesis addresses the issue of automating abstraction reformulations for computational efficiency.

2.6 Automating Reformulation

> *In any knowledge base, and for any intelligent agent, it is essential to make the right distinctions in order to be able to organize and cope with the complexities of the real world. In order to know what constitutes a good set of individuals, categories, attributes and relations, we have to understand how the possession of for example, a category, in one's vocabulary assists in making appropriate decisions.*
>
> from Lenat et al (eds), *The Ontological Engineer's Handbook, 2nd ed.*, Addison-Wesley, 1997, p. 1.

There are two parts to the reformulation problem: the epistemological part and the heuristic part. The epistemological part is concerned purely with what constitutes a correct reformulation of a problem, i.e. what the space of reformulations is. The heuristic part is concerned with how we can actually generate reformulations. The solution to the epistemological part is a proposal for a generator of reformulations, the heuristic part addresses the question of how to tame this generator. Both parts are important for a satisfactory solution to the problem, but the epistemological part has to be resolved before the heuristic aspects can be tackled. The role of an epistemological analysis of reformulation is to tell us what a reformulation is, and what the logical character of a reformulation inference is.

2.6.1 Formulating the Automation Problem

Given that reformulation is the process of changing distinctions in the world, and that it is achieved by changing theories, we can now frame the automation question for deductive abstraction reformulations.
Given

- An initial conceptualization $\mathcal{C}_1$ and its encoding E_1
- Goals G (correctness constraints)
- Problem solver PS: description of behaviour + cost model
- Computational constraints S on solution of goal by PS(goodness constraints)

Produce a new abstract conceptualization $\mathcal{C}_2$ and implement it in the encoding E_2 that meets correctness and goodness constraints.

2.6.2 Previous Work

Much of the earlier work on reformulation has been of an exploratory nature. The most influential piece of research was that of Saul Amarel in 1968 [Ama68]. Amarel presented examples of reconceptualizations and re-encodings for computational efficiency in the missionaries and cannibals puzzle. He also speculated on methods for their mechanization. One important shift in this puzzle is the abstraction of the named individuals into appropriate sets. It set the stage for the creation of abstract action operators that make the solution of the problem very efficient. This reformulation can now be automated by general methods proposed in this thesis.

The difficulties of automating reformulation deterred research in this area for a long while. In 1980, Korf [Kor80] attempted to characterize the nature of the information-processing that occurs during special-purpose reformulations. He defined a set of rewrite

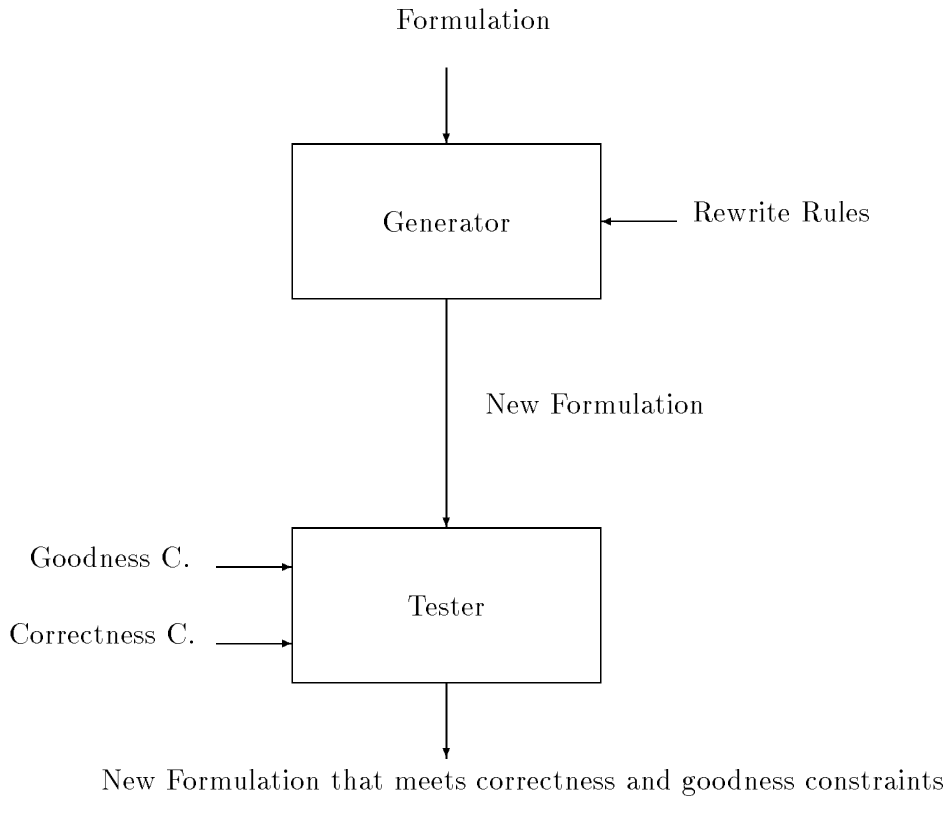


Figure 2.10: The Transformational Approach to Reformulation

rules on encodings that set up a space of possible re-representations. His theory explained representation shifts in the Tower of Hanoi puzzle as well as some examples from floor planning. In 1981, Jack Mostow designed a program that reformulated advice in the game of Hearts: he devised 400 transformation rules to generate the space of reformulations. These transformation rules were a set of relational rewrite rules of the form: $exp1 \rightarrow exp2$ and they were used to generate new encodings from the given one. These rules formed the generator for the space of encodings and the computational constraints were used as testers to determine if an appropriate encoding E_2 had been found. The architecture of this solution is sketched in Figure 2.10. The main disadvantage of this approach was that there were too many rewrite rules even in domains of medium complexity (Hearts) and the control problems were formidable. They were solved by having the user guide the reformulation process. Another major disadvantage of the approach was that there was no principled way of actually generating these rewrite rules from more global considerations.

In 1983, Utgoff [Utg86] attacked the inductive reformulation problem in the context of the LeX [MTMB82] problem solver. He proposed a method called *back-propagation* to refine the concept of integer to *odd-integer* to make an integration heuristic expressible. The chief insight in this work was the explicit use of problem-solving goals to guide the addition of new concepts. The addition of new concepts to an inductive system to make prediction efficient was studied by Fu & Buchanan [FB85]. This work proposed two general heuristics and was tested empirically in the domain of medical diagnosis. In 1986, Richard Keller [Kel87] introduced a scheme for the addition of new concepts into the LeX problem solver that would make the solution of some queries very efficient. He used a sophisticated model of problem solving and empirical methods for testing the efficacy of a given representation against a class of goals.

More recently, there has been a flurry of work on the problem of automatic introduction of new vocabulary to make problem solving and machine learning more efficient. Most of them focus on very special classes of deductive and inductive abstractions. All of them pose the reformulation question at the level of encodings. Patricia Riddle [Rid88] under Saul Amarel has attempted to automate the formation of deductive abstractions of problems formulated in the state-space framework. She has focussed on the creation of macro-operators that speed up the solution of a class of queries. Mike Lowry [Low88] has developed abstraction methods based upon the theory of abstract data types to synthesize algorithms from specifications. Muggleton [Mug88] has devised a machine learning scheme that introduces new predicates to make the resulting generalization compact. Flann [Fla88] reformulates theories expressed in structural terms into functional terms to make a recognition task more efficient. The novel aspect of this work is the use of examples to guide selection of useful functional terms. The generation of functionally useful classes by induction from problem solving traces is also the theme in Jeff Schlimmer's work on representation change [SP88].

2.6.3 The Justification Based Approach

Our credo is that automation of reformulations is not possible unless the reformulator can *justify* a shift in conceptualization. The hope is that if the justification or explanation of a reformulation is done at the right level we[5] can exploit it to automate the process. Standard explanations for the correctness and goodness of a reformulation are at too low a level of detail; they obscure the important aspects of the proof that are needed for generation. This is the same insight found in the analysis by synthesis work in the domain of electrical circuits by Sussman[Sus77]. A circuit can be synthesized by writing down the equations for behaviour in terms of the unknown values and solving for them. This usually results in huge systems of equations. However, we can use knowledge of the form of the answer to guide us in setting up the equations cleverly, so they can be easily inverted. As Sussman puts it: it is knowing what algebra to do that separates a good circuit designer from a bad one. The explanation framework introduced in this thesis provides a way of doing algebra on formulations in a clever way.

The justification based approach involves asking the questions: *Why is the conceptualization $\mathcal{C}_2$ a reformulation of $\mathcal{C}_1$?. Why is E_2 a reformulation of E_1?* These are explanation-seeking why questions in Hempel's [Hem65] terms. Our object in doing this is to articulate the knowledge and the reasoning that goes into deliberate reformulation so that we can compile it into algorithms for automatic reformulation. But what does an explanation or a justification for a reformulation look like?

Reformulation can be defined as the process of inferring a new conceptualization and a new theory in that conceptualization that preserves the goal and that satisfies the given effectiveness criteria S with respect to a problem solver PS.

[5] as designers of automated reformulators

$\boxed{\text{From } \Delta,\ \mathcal{C}_1,\ E_1,\ g_1, S \text{ infer } \mathcal{C}_2,\ E_2,\ g_2,}$

This is a non-deductive argument in that the conclusions do not follow syntactically from the premises. The justification problem then is to find a criterion, which if satisfied by a reformulation inference, sufficiently establishes the truth of the conclusions. Our objective is to find general principles which when taken with background knowledge and added to the premises of the reformulation inference, make the conclusion follow soundly.

The goal of this research is to provide a normative justification for a reformulation inference and in doing so provide a general form for the background knowledge needed to draw reformulation conclusions regardless of the specific method used to derive them. That is, we are interested in enumerating the space of reformulation conclusions starting from an initial set of premises. Some criteria on the nature of this justification knowledge are:

- Content
  The justification should be a declarative statement of the factors that come into play in the choice of formulation: what the semantics of that choice are and what role they play in the problem solving process. Much of this knowledge is left implicit in present day systems, so when computational constraints are changed, a system cannot realign conceptual boundaries in a knowledge-based way.

- Generality
  The justifications should be domain and problem-solver independent. This does not mean that they will be insensitive to such knowledge, it simply necessitates factoring out as much of the domain and problem-solver specific information as possible to facilitate reuse.

- Generative power
  The justification should be structured in such a way that it can be used to *generate* new formulations. This is akin to the technique of synthesis by analysis in analog circuit design [Van74]. The justifications then, are not purely explanatory in nature; they can be *run in reverse*[6] to suggest the space of possible reformulations that satisfy the correctness and goodness constraints.

[6] as suggested by Sussman in his work on slices

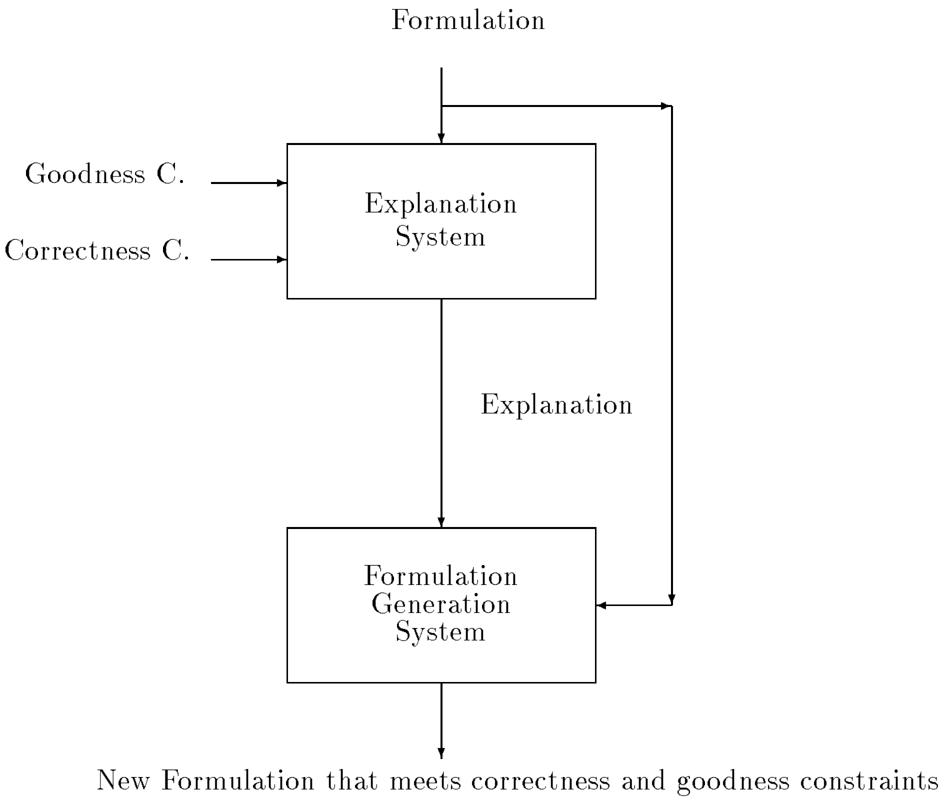


Figure 2.11: The Justification Based Approach

$$\begin{tikzcd}
\text{Irrelevance Claims} \arrow[r, "\text{Irrelevance}", "\text{Principle}"'] & \emptyset \\
\text{Formulation 1} \arrow[u] \arrow[r, "\text{Reduction}", "\text{Inference}"'] & \text{Formulation 2} \arrow[u]
\end{tikzcd}$$

Figure 2.12: The Relevance of Irrelevance

Justifying *changes* in conceptualization requires that we be able to justify conceptualizations in the first place. A conceptualization partitions the universe in a certain way. A justification for a conceptualization articulates the epistemological and computational consequences of assuming those distinctions. A justification for a *change* of conceptualization is an explanation for the introduction or removal of some conceptual elements for the achievement of the given goals with new resource constraints. These justifications are thus meta-theoretical. They attempt to answer why the current formulation fails to meet the specified computational constraints.

Justifications for reformulations that satisfy Definition 12 have two parts to them: the *correctness proof* that guarantees that the new formulation obtains the same answers on the given set of goals and the *goodness proof* that shows that the new formulation has better computational properties modulo a given problem solver. These explanations invoke properties of the current conceptualization as well as the present encoding to explain the epistemic and computational role of the distinctions being made. The advantage of these meta-theoretical justifications is that they tie the change in formulation directly to a change in computational properties. One class of such explanations are *irrelevance justifications*. We then use our knowledge of conceptualizations, encodings, and the problem solver to redesign the formulation to meet the new constraints.

2.6.4 Irrelevance Justifications

One of the simplest roles a distinction could play is that it has no part or has a dispensible part in the computation of a set of queries. We then say that the vocabulary term is irrelevant to the particular set of queries. An irrelevance justification for an abstraction reformulation explains why some conceptual elements were expendable and why some distinctions can be collapsed to get more abstract terms. The *irrelevance principle* states that a formulation should be changed to eliminate *all* distinctions irrelevant to the present goals. It thus sanctions the inferences that logically minimize distinctions made in a formulation. Thus, we can integrate momentary objects into appropriate intervals as well as individual points in space into lines. The compression of particulars into a universal, the subject of machine learning, is also governed by the same principle: generalization occurs because we discard distinctions that do not have predictive utility. At the meta-theoretic level, the irrelevance principle sanctions a move towards conceptualizations and encodings that make no irrelevant distinctions as in Figure 2.12. The principle simplifies computation in a theory by making the objects assumed by it as few and as large as is consistent with the correctness constraints. This ontological economy in describing problems entails computational savings. We only make distinctions necessary for the purpose at hand. This informal idea expounded in [Qui63, Har86] among others is made precise in the succeeding chapters so that we can design a machine that obeys this principle.

Chapter 3

The Theory of Irrelevance

3.1 Introduction

Often, wisdom is knowing what to ignore. An autonomous resource-limited agent with a very detailed theory of the world should be able to reformulate it to a simpler theory that allows it to make predictions at the required level of accuracy within the given resource constraints. Such an agent has to identify distinctions made in its conceptualization of the world that are irrelevant to the class of predictions it is designed to make, and weaken its theory by removing irrelevant distinctions. The theory of irrelevance provides a logical basis for justified discarding and ignoring of information and the construction of computationally effective theories from detailed, intractable ones.

3.2 Motivations

There is too much information in the world and an intelligent agent has to focus selectively on it and structure it in effective ways to achieve its goals. An agent thus needs to reason about what information can be ignored and why. Removing irrelevant information has important consequences, both computational and epistemic. In Abstrips [Sac74], the ignoring of preconditions of lower criticality while attempting to achieve an abstract plan at a certain criticality level, leads to overall efficiency in the planning process. In case-based expert systems for medical diagnosis, the introduction of additional evidence often degrades accuracy of performance because the number of spurious matches increases. The quality of answers and explanations obtainable from a knowledge-based system is improved if these systems are endowed with the ability to explicitly reason about what to ignore in a goal-sensitive way.

The theory of irrelevance is a tool for specifying as well as deriving classes of information that can be ignored in the context of particular goals. For instance, in Dendral, there are two classes of mass spectrograph points that are ignored for the purposes of the structure interpretation task. Scientists have much sharper criteria for the data points to ignore rather than the data points to include. *Specification of irrelevance claims is a valuable mode of expressing knowledge about a domain.*

Another important reason for building the ability to reason about irrelevance into systems is that we would like to give advice about irrelevance of entities to a problem-solving system. For example, in the missionaries and cannibals problem, we would like to tell the problem-solver that the names of the missionaries and cannibals are irrelevant, and have the system clump the missionaries and cannibals into sets and reason with the cardinalities of these sets. Amarel indicates this sort of reasoning in his well-known 1968 paper [Ama68]. *Removing irrelevant facts and objects from a formulation is an important method of changing representations.* Yet another motivation for reasoning about irrelevance is the need for problem-solving systems to reason flexibly at varying grain sizes [Hob85]. These systems require the ability *to recognize and ignore detail irrelevant to their current goals* in order to shift to a bigger grain size where those goals can be achieved more efficiently.

Reasoning about irrelevance is equally important in learning and theory formation. Minsky explains the role of irrelevance in learning very powerfully in the following excerpt from the The Society of Mind.

> ...... *we never ever face the same appearance of anything twice. We are almost certain the next time to be looking from a different viewpoint, nearer or further, higher or lower, in a different colour or against a different background. So unless our minds simplify away the inessential aspects of each scene, we could never learn anything.*

Reasoning about irrelevance can thus be used as a *basis for focusing attention in both inductive and deductive tasks.* In induction, irrelevance claims bias the learner towards the construction of simpler and computationally more effective generalizations. In deductive tasks, irrelevance statements help focus search by identifying unfruitful or redundant paths. They also help *restructure the search space* by introducing new primitives.

3.2.1 Guide to Chapter

To this end, this chapter (which is an extension of [SG87]) introduces meta-level statements about irrelevance that allow us to specify which distinctions in a formulation can be dispensed with on logical or utilitarian grounds with respect to a given task. We then outline the semantics and properties of statements about irrelevance. We present a theoretical framework for irrelevance and develop a hierarchy of logics that capture different senses of irrelevance. The logics of irrelevance serve as a language for specifying irrelevance claims in the world and their associated calculi allow us to draw new irrelevance conclusions from given ones. We present proofs of irrelevance in the logics. Efficient graph-theoretic compilations of some of the calculi are also given.

3.3 Informal Semantics of Irrelevance

A fact f is irrelevant to the goal schema g in the context of a set of sentences T, written as *Irrelevant*(f,g,T), if perturbing the value of f in T does not affect that of g. Informally, the following conceptual derivative is calculated.

$$\boxed{Irrelevant(\mathsf{f}, \mathsf{g}, \mathsf{T}) \equiv (\tfrac{\Delta \mathsf{g}}{\Delta \mathsf{f}} \mid_{\mathsf{T}} = 0)}$$

This is an exact irrelevance claim. Approximate irrelevance claims are those in which the above derivative does not equal zero, but some ϵ very close to zero.

Now we present some examples.

1. **The Price of Tea in China**

Given our current knowledge of economic theory, the price of tea in China is irrelevant to my writing this thesis. This is an exact irrelevance claim. Even if we changed the value of the price of tea in China in our theory of the world, the change would not propagate to the fact about my writing this thesis. The irrelevance claim is a fact about the relationship between two facts in our theory of the world. There are two possible meanings to this meta-relation. One, even if we simplified our theory of the world by discarding information about the price of tea in China, the truth value of the proposition about my writing this thesis would not be affected. This is the *subtractive semantics* of irrelevance. Its counterpart is the *additive semantics* that says that even if we added information about the price of tea in China, it would not help us conclude anything more about my thesis writing. Since the relations Price-of-Tea and Write-Thesis are logically independent in our theory of the world, these two interpretations of irrelevance are identical.

2. **The Hybrid-pi model**

As an example of an approximate irrelevance claim consider the following. For the purposes of computing the low frequency gain of a transistor using the hybrid-pi model, the base-emitter and base-collector capacitances are irrelevant. These two components complicate the analysis of the circuit without providing a commensurate increase in accuracy for the low frequency case. Under these conditions, we would like to ignore the effects of the capacitances and simplify the model by dropping them. In the previous example, the elimination of the Price-of-tea relation did not influence the truth of the proposition about my thesis; here removing the capacitors causes the value of the gain to change, but not significantly.

Notice that the irrelevance claim about transistors is a conditional one, it is true only for frequencies less than 50 Hz. Also, unlike the previous example, an *object* (a base-emitter capacitor in the hybrid-pi model) is specified as irrelevant. We now consider what it *means* for an object to be irrelevant. A subtractive semantics for object irrelevance is: even if we removed the object from the model of the theory, we would still be able to solve for the goal. An additive semantics for object irrelevance is: addition of the object does not tell us any more about the goal (in particular, it does not make a previously unsolvable goal solvable). Object irrelevance can be treated as a special case of fact irrelevance; f would then be the statement that an object with the requisite properties exists.

The irrelevance claim above can be determined by a meta-theoretical analysis of the equations for calculating the gain using the hybrid-pi model. Under the low frequency condition, the capacitive terms are second-order effects: an order of magnitude reasoning over the various terms that contribute to the gain shows that the capacitive terms have a negligible effect. Proving irrelevance claims is a creative endeavour: in this case the proof requires doing a sensitivity analysis of the gain equations.

Removing an object from the model entails modifying the theory so that the existence of the object can no longer be deduced. In model-theoretic terms, this entails modifying the Herbrand base to exclude the object. In terms of revising the theory, this amounts, in the simplest case, to removing all references to the object in the theory (akin to dead code elimination in compiler optimizations). In our example above, we can simplify the gain equations by this method. In more complicated cases, we have to remove those facts that assert that the object exists as well as those that depend on the fact that the object exists. We continue with more examples.

3. **Missionaries and Cannibals**

In the missionaries and cannibals puzzle, the solution does not ask for a particular individual to reach the destination bank ahead of another. In the 3×3 problem, there are 36 possible solutions that differ on the *order* of the missionaries and cannibals reaching their destination. This number is derived by using the fact that every permutation of the missonaries amongst themselves as well as the cannibals amongst themselves transforms a valid solution into another valid solution. The fact that the order is irrelevant, allows us to erase the identities of the individuals and clump them into sets. On analyzing the preconditions of the action operations in this puzzle (load-boat-left, move-boat-from-left-to-right, etc.), we see that the preconditions only require the cardinalities of the sets of individuals in each bank and the boat. This gives us the justification to discard all attributes of the sets except for their cardinality.

Suppose we introduced two new operators Sit-Down and Stand-Up that act on individuals. Recall that the goal is to find the minimal sequence of actions that achieves the transfer of the missionaries and the cannibals. We can state the fact that no minimal sequence of actions to achieve the goal uses these two operators by making the claim that these operators are irrelevant to the goal. This means, even if the operators are removed, the same solution would obtain. The irrelevance claim captures an important property of the problem space intensionally . This particular irrelevance claim can be discovered by a local analysis of the preconditions of operators in time linear in the number of operators.

4. **Abstrips**

The essence of the Abstrips approach for controlling search in planning is to use a means for distinguishing details from essential aspects of the problem space [Sac74]. By planning in a hierarchy of abstract problem spaces ordered by the amount of detail in them, and by introducing detail in a top-down fashion, a search space which is exponential in the number of operators is reduced to one that is polynomial in complexity.

In contrast with all the irrelevance claims above, the claims in Abstrips cannot be deduced from a description of the most detailed space. The hierarchy embodies knowledge about what is detail and what isn't at particular points in the planning process, and this knowledge[1] is expressed in the irrelevance claims. By changing the criteria for what

[1] which is not expressed in the detailed domain theory!

constitutes detail, Abstrips can construct abstraction spaces that are computationally useful for the task at hand.

For instance, the **Turn on the lamp** operator is represented in STRIPS notation [Sac74].

$$
\begin{aligned}
\text{Preconditions :}\quad & \mathsf{Type}(l, LAMP) \wedge \exists rx. && (1)\\
& \mathsf{Inroom}(ME, rx) \wedge \mathsf{Inroom}(l, rx) && (2)\\
& \mathsf{Plugged} - \mathsf{In}(l) \wedge && (3)\\
& \mathsf{Nextto}(ME, l) && (4)\\
\text{Addlist :}\quad & \mathsf{On}(l) &&\\
\text{Deletelist :}\quad & \mathsf{Off}(l) &&
\end{aligned}
$$

The predicates in the preconditions are ordered in decreasing order of criticality. At criticality 1, the predicates of lower criticality (**Inroom**, ..., **Nextto**) are ignored. Translating this into state space terms, all states that differ purely on the values of the predicates of lower criticality are treated as identical. This is the assertional import of the criticality assignment; it clumps whole subgraphs in the state space graph into a single node.

The information contained in the criticality assignments can be given a clean semantics by re-expressing them as irrelevance claims. This also allows us to declaratively specify the criterion of irrelevance which can change from task to task and permits the automatic derivation of the required abstraction hierarchy. Abstrips uses the following (procedurally expressed) criterion to sift detail from the essentials. If a predicate (that expresses a condition of the world) is easier to achieve than another, it deems the first predicate to be an irrelevant detail and drops it from consideration.

To formalize the above, we need to enrich our simple definition of irrelevance to include an ordering criterion. If we have two fact schemas f_1 and f_2 that are irrelevant to the same goal-schema g, we can impose the "Is-More-Irrelevant-Than" ordering on f_1 and f_2. To automate the construction of the particular hierachy that the designers of Abstrips chose, we first define the semantics of this ordering to be ease of achievability. We then construct the abstraction hierarchy bottom up by dropping the least element of this partial order first, and proceed iteratively, until all the elements in the order are covered.

For example, suppose we have a theory T with two predicates f_1 and f_2, and we also know that in the context of achieving goal schema g, f_1 is more irrelevant than f_2. Then the irrelevance minimization method would construct a 2 level abstraction hierarchy whose top level is the theory T_1 which ignores f_1, and whose bottom level contains T. In effect, we perform the following inference.

Description of detailed space + Specification of Is-More-Irrelevant-Than claims for a given class of problems $\overset{MinIrr}{\Longrightarrow}$ *The Abstraction Hierarchy.*

The irrelevance minimizer guarantees that at each level the abstract theory ignores detail that is irrelevant for that level as well as the levels above it.

5. **Founding Fathers**

Consider the following kinship example. We start with trees of Father relations. The defined relation Ancestor is the transitive closure of the Father relation. The goal is the SameFamily relation; two people belong to the same family if they have a common ancestor. If a simple backward chaining system like Prolog worked on this formulation of the problem, SameFamily queries would take time proportional to the height of the father tree.[2] Suppose we wish to reformulate this problem in order to be able to solve for SameFamily queries in constant time with O(n) extra space where n is the number of people in the family trees.

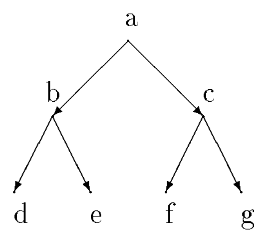
There are two irrelevance claims that allow us to accomplish the reformulation.

1. The distinction between immediate and non-immediate ancestry (i.e., between Father and Ancestor) is irrelevant to the SameFamily query.

2. The identity of the common ancestor is irrelevant to the SameFamily query.

While the irrelevance claims in Abstrips, collapsed states in the state space into equivalence classes modulo the plan to be achieved at a certain level of abstraction, the claims in this example identify redundant paths in the search space. If the Ancestor fact corresponding to a given ground Father fact were available in the formulation, there would be two alternate ways of concluding SameFamily: one that uses Father(x,y) $\Longrightarrow$ Ancestor(x,y) and the ground Father fact, the other that terminates on the ground Ancestor fact. The irrelevance claims sanctions the construction of a weakening of the formulation by dropping the Father relation. This is equivalent to relabelling the Father trees as the Ancestor trees in the formulation. Note that model-theoretically, we get rid of the distinction between Ancestor and Father. Proof-theoretically, all proofs of Same-Family are shortened by one step. And in terms of the search space, this relabelling causes all branches that result from expanding the first axiom in Figure 3.1 to be pruned. The irrelevance claim has a computational impact. To capture this notion precisely, we refine the notion of irrelevance introduced in the beginning of the chapter as: f is computationally irrelevant to g

[2] ... if they succeed. If not, the interpreter generates an infinite subgoal sequence.



Father(a,b)
Father(a,c)
Father(b,d)
Father(b,e)
Father(c,f)
Father(c,g)
Father(x,y) $\Longrightarrow$ Ancestor(x,y)
Ancestor(x,z) $\wedge$ Ancestor(z,y) $\Longrightarrow$ Ancestor(x,y)
Ancestor(z,x) $\wedge$ Ancestor(z,y) $\Longrightarrow$ SameFamily(x,y)

Figure 3.1: The Given Formulation

in the context of T if the conceptual derivative of g with respect to f in T is zero *and* the simplification of T that is constructed has better computational properties.

The second irrelevance claim has a stronger computational impact. It sanctions the inference that allows us to shortcircuit all the Ancestor links upto the roots of the trees. In Section 3.5 we will state this irrelevance claim very precisely. This claim leads to the synthesis of the *maximal Ancestor* relation. The mechanics of the theory revision that accomplishes it is the subject of Chapter 4. The abstract justification for both these claims is: a fact schema is irrelevant if there exists an alternate fact schema that reaches the solution without it.

6. **Granularity Shifts**

The width of the road is irrelevant to the length of the path from San Fransisco to Los Angeles. This means that two points that differ only on the width dimension along Rte 101 are in the same equivalence class with respect to distance from Los Angeles. Model-theoretically, this allows us to project out the width component from the Herbrand base of a theory that treats the road as a surface. Points on the road in this theory are regarded as elements in $\mathcal{R}^2$ and the irrelevance claim sanctions the simplification to a theory whose Herbrand base contains the projection along one coordinate which are elements in $\mathcal{R}$. Computationally, this is a win because we have to compute the length of the *line* joining

San Fransisco and Los Angeles as opposed to computing the average of such lines on the *surface* connecting San Fransisco to Los Angeles.

Another example is the coarsening of a discretization: e.g. we may wish to reformulate a theory which distinguishes between temperatures in the interval [58..60] to one that does not. This requires treating distinct objects 58,59,60 as elements of a single equivalence class. The theory in which they occur is modified so that the fact that $58 \neq 59 \neq 60$ can no longer be deduced in it.

7. **Abstraction in Digital Circuits**

We present an example of structural abstraction. The idea behind this is that the specification of the device should not reflect its internal structure, but only its externally observable behaviour. Suppose we have the gate level description of a full-adder circuit as in Figure 2.1. Suppose further, that we deem the exact values of the signal at the points d, e and f irrelevant. The current theory of the full adder allows us to deduce the output values at the above mentioned points in the circuit. However, all we care about is that there *exists* a value. We therefore need to weaken the theory to make these values undeducible. One minimal weakening that accomplishes this without changing the values of sum and carry, changes the Herbrand base. It introduces a new object called the Full-Adder FA as in Figure 2.2 which has 3 inputs, a, b and c and outputs sum and $carry$. We compress the reasoning chains through d, e, and f in the detailed theory and express the values of sum and $carry$ entirely in terms of a, b, and c. By changing the assignment of points in the circuit that we wish to make irrelevant, we can segment the circuit in goal-sensitive ways to get good abstractions.

8. **Macrops**

The circuit example above as well as the kinship example are instances of formation of macroperators in the search space by the elimination of irrelevant intermediate variables (circuit points in the former and ancestors in the latter). The ignoring of intermediate states leads to the formation of macroperators and the criterion for ignoring them can be naturally expressed as irrelevance claims. Macrops are a very fertile area of research in machine learning [REFJ81, Kor83]. The difference between the macrops constructed in this thesis and the standard ones in literature is that not only do we reconfigure the search space by eliminating intermediate steps, we also propagate the change into the formulation so that the new formulation does not generate those irrelevant steps.

9. **Selective Forgetting**

Resource limited agents have limited memory and have to decide which information to discard and which to keep. Policies for selective retention of information or forgetting can be expressed as irrelevance claims. For instance, an agent that lives in a world in which information obtained at time t becomes irrelevant with respect to its goals at $t+2$ could be told it as an irrelevance claim. This would sanction an inference on the part of the agent to ignore information more than 2 clock ticks old. Throwing away information can be computationally beneficial in other contexts. In macro-operator formation in a machine learning system, the accretion of macros slows the system down [FN71, Min88]. The rationale for discarding macros can be phrased in terms of the utility-theoretic version of irrelevance, called computational irrelevance.

10. **Control Reformulations**

Here the domain of discourse is over computational objects like proof trees and computation sequences and the irrelevance claims state the fact that there is wasted computation going on. Consider the following formulation of the Fibonacci function:

$$\mathsf{Fib}(n) = \begin{cases} 1 & \text{if } n = 1; \\ 1 & \text{if } n = 2; \\ \mathsf{Fib}(n-1) + \mathsf{Fib}(n-2) & \text{if } n > 2. \end{cases}$$

The sequence of subgoals generated by a backward-chaining system like Prolog is:

$\mathsf{Fib}(5)$

$\mathsf{Fib}(4)\mathsf{Fib}(3)$

$\mathsf{Fib}(3)\mathsf{Fib}(2)\mathsf{Fib}(3)$

$\mathsf{Fib}(2)\mathsf{Fib}(1)\mathsf{Fib}(2)\mathsf{Fib}(3)$

$\mathsf{Fib}(2)\mathsf{Fib}(1)\mathsf{Fib}(2)\mathsf{Fib}(2)\mathsf{Fib}(1)$

If $\mathsf{Fib}(n)$ has already been computed before and has been stored away, it is irrelevant to compute it again using the definition. This is the structure of the explanation for reformulations that eliminate repeated computation. If the objective is to minimize the number of compute actions, then compiling this irrelevance claim into the formulation requires rewriting it so that a Fib value is looked up rather than computed whenever possible.The new relation introduced by the irrelevance minimizer caches a part of the computation sequence (e.g. the previous two Fib values at any point in the Fib computation).

The irrelevance claim in this case is a conditional one that says that if there is a stored value for a Fibonacci computation, then the computation of that value in the context of any goal is irrelevant. Notice that what is deemed irrelevant in this case is the *computation* of $\mathsf{Fib}(n)$ from the definition. We can express the meaning of this claim in terms of the fact based semantics introduced informally in Section 3. To do this we need to distinguish *computed* Fib's from *stored* Fib's.

This reformulation is called a control reformulation because the entities that are deemed irrelevant are computations. Also the effect of the reformulation, viz., the minimization of repeated computation, can be achieved by changing the problem solver while keeping the formulation intact. Forward chaining from $\mathsf{Fib}(1)$ and $\mathsf{Fib}(2)$ on the current formulation until we obtain the required Fib value is an optimal computation. Control reformulations can be described by the following equation.

$$\boxed{F_1 + PS_1 = \begin{cases} F_2 + PS_1 & \textit{An irrelevance reformulation;} \\ F_1 + PS_2; & \textit{Meta-level Control of Inference} \end{cases}}$$

where F_1 and F_2 stand for the current and the new formulations respectively and PS_1 and PS_2 are the given and the modified problem solver. Control reformulations move knowledge from the interpreter to the formulation itself. They are thus information-gaining, since the new formulation has compiled control information that was absent in the old formulation.

3.4 Taxonomy of Irrelevance

There are atleast two axes on which to taxonomize irrelevance : the type of the entity being deemed irrelevant, and the sense in which that entity is irrelevant to a goal. The irrelevance claims in the Fibonacci case referred to wasted computation, the claims in the kinship example referred to facts that could be dispensed with. For each type of f, we can define the perturbation $\Delta\mathsf{f}$ that is legal. The irrelevance claims in the PriceofTea case and the Hybrid-π case differ on the exactness of irrelevance: the exact claims ($\frac{\Delta g}{\Delta f}$ = 0) are *logical* irrelevance claims. Some logical claims are *approximate*. An approximate irrelevance claim asserts that ignoring the entity deemed irrelevant is a computationally beneficial thing to do (irrespective of whether or not is logically correct to do so). The type of perturbations we are willing to consider include

1. f is a proposition : flip the value of f in the theory

2. f is a set of propositions : flip the value of any f in that set

3. f is an object : remove f from the Herbrand base of that theory
4. f is a set of objects: remove them all from the Herbrand base
5. f is part of a proof tree or search space : prune that part of proof tree or search space
6. f is an action: do not perform action
7. f is an argument of a relation: drop that argument

Irrelevance claims can be **probabilistic** too. This involves attaching probabilities to the claims: most independence claims in the physical world as well as irrelevance claims in ill-structured domains like medicine have this flavour. Probabilistic claims will be studied in the future. In this thesis we limit ourselves to exact irrelevance claims. These are of two kinds: logical and computational. A logical irrelevance claim allows the discarding of information and guarantees that the answers to the given set of goals will be unaffected. An approximate irrelevance claim like the hybrid-π case, allows the discarding of information but does not guarantee that the answers to the given set of goals will remain the same. A special class of logical irrelevance claims are called **computational irrelevance claims** because not only do they guarantee that discarding some information preserves the answers to some goals of interest, but that throwing away some information causes the computation of the goals to proceed faster.

3.5 Formalizing Irrelevance

Inefficient formulations make irrelevant distinctions. An irrelevance claim expresses a justification for an abstraction reformulation. It explains why some conceptual elements in a formulation are expendable and why some distinctions can be collapsed to get more abstract ones. We devise a logic of irrelevance to state irrelevance claims and to derive them mechanically from the given formulation. We do so by introducing a relation *Irrelevant* that takes 3 arguments: f: the thing that is deemed irrelevant, g: the goal and T: the theory or context in which irrelevance is established.

3.5.1 The need for a meta-theoretical analysis

If we were asked whether or not, $x^{999} + x^{888} + x^{777} + \ldots + x^{111}$ is divisible by $x^9 + x^8 + x^7 + \ldots + x^1$, we have two choices;

1. Perform the actual division. This is a long and computationally expensive process which results in the generation of the quotient.

2. Utilize the fact that the quotient was not needed at all, all we wanted was to determine whether or not the polynomial was divisible. There are sufficient conditions for divisibility that we can use to determine this property that use easily evaluable conditions on the forms of the divisor and dividend.

The same motivation underlies the logic of irrelevance; it gives us a way to express and use information about dependencies between sets of facts in a formulation without carrying out the detailed proofs. The logic of irrelevance allows the succint expression of the fact that some distinctions can be dispensed with on logical or utilitarian grounds, it also justifies the discarding of distinctions in the form of a proof of irrelevance. The logic of irrelevance essentially performs a qualitative analysis of proofs of a class of goals in a formulation – it makes possible intensional reasoning about proofs and about formulations in which they arise, without exhaustively enumerating them. When the domain of discourse is search spaces, the logic of irrelevance allows us to intensionally reason about pruning search paths.

3.6 The propositional logic of irrelevance

Here the arguments of the ternary relation *Irrelevant* are restricted to be propositions or sets of propositions. We say that f is irrelevant to g in the context of the set of sentences T, written as $Irrelevant(\mathsf{f}, \mathsf{g}, \mathsf{T})$, if changing the truth value of f in T does not affect that of g. The truth value of f in T is changed by constructing a weakening T' of T that no longer supports the truth of f. First, we define a weakening.

Definition 14 *The set T' of sentences in the vocabulary $\mathcal{L}$ is weaker than the set T in the same vocabulary, if $\mathsf{T} \models \mathsf{T}'$ or equivalently if Deductive-Closure(T') $\subset$ Deductive-Closure(T).*

Two classes of weakenings are explored in this thesis. Both use the subset operation. The first class, called Type 1 weakenings decrease the deductive closure of the set without changing the Herbrand universe. Type 1 weakenings are characterized by Definition 1. Type 2 weakenings collapse propositions by constructing equivalence classes of them and thus change the Herbrand base. We study Type 1 weakenings only. The irrelevance claims described are called *weak* irrelevance claims.

Definition 15 *WI_1(f,g,T) $\equiv$ T $\models$ g and $\exists$ T$'$. such that T$'$ is a weakening of T and T$' \models$ g and T$' \not\models$ f.*

Example 1 *Let $\mathsf{T}_1 = \{\mathsf{l}, \mathsf{t}\}$. Let $\mathsf{T}_2 = \{\mathsf{l}, \mathsf{l} \Rightarrow \mathsf{t}\}$ where l and t stand for the propositions that there is lightning and thunder respectively.*

Using Definition 15, we can conclude that $WI_1(\mathsf{l}, \mathsf{t}, \mathsf{T}_1)$ since we can construct a T' with the required properties. However, we cannot show that $WI_1(\mathsf{l}, \mathsf{t}, \mathsf{T}_2)$. This is because the truth of t in T_2 does depend on that of l, so the conceptual derivative $\frac{\Delta \mathsf{t}}{\Delta \mathsf{l}}$ is not 0 in T_2.

This example shows a very interesting property of WI_1. Even though T_1 and T_2 are identical from a model-theoretic viewpoint, the irrelevance judgements in the two sets are not. Claims about WI_1 are sensitive to the form of T. This precludes describing the semantics of WI_1 using model theory alone. The status of t in the two sets T_1 and T_2 is different; in T_1 it is a primitive proposition (its justification set is empty), whereas in T_2 it is a derived proposition (its justification set is the singleton containing l). This distinction between primitive and derived t's is lost in the model theoretic accounts of the meaning of the two sets.

This example brings up an interesting kind of non-monotonicity that WI_1 possesses. The irrelevance claims are not preserved across the deductive-closure operation on T.

Example 2 *Let $\mathsf{T}_1 = \{\mathsf{p}, \mathsf{p} \Rightarrow \mathsf{q}\}$ and let $\mathsf{T}_2 = \{\mathsf{p}, \mathsf{p} \Rightarrow \mathsf{q}, \mathsf{q}\}$.*

It is the case that $\neg WI_1(\mathsf{p}, \mathsf{q}, \mathsf{T}_1)$ and $WI_1(\mathsf{p}, \mathsf{q}, \mathsf{T}_2)$ even though T_2 is generated from T_1 by adding an entailed conclusion. So irrelevance claims of this form are not preserved across addition of deductively entailed conclusions. However, note that the conditional irrelevance claim that holds in both sets is $\mathsf{q} \in \mathsf{T} \Rightarrow WI_1(\mathsf{p}, \mathsf{q}, \mathsf{T})$. This states that if q were present in the set T, then p would be irrelevant to q. For T_2 the LHS is true and thus we get $WI_1(\mathsf{p}, \mathsf{q}, \mathsf{T}_2)$. For T_1, we interpret the condition, counterfactually – if q were added to T_1, then p would be WI_1 to q.

Definition 15 does not usually specify a unique T'. If there are multiple subsets of T such that they entail g without entailing f, then we have to choose which one will be constructed by our automated irrelevance reasoner.

Definition 16 *$WI_{max}(\mathsf{f},\mathsf{g},\mathsf{T}) \equiv \mathsf{T} \models \mathsf{g}$ and $\exists\, \mathsf{T}'.\ \mathsf{T}' \models \mathsf{g}$ and $\mathsf{T}' \not\models \mathsf{f}$. where T' is a maximal subset of T with this property.*

One might wonder why we impose the restriction of maximality on T' in this definition. This requirement is not essential for the determination of irrelevance, it is necessary however, that the revision of T that we construct be the most conservative one, so that we do not lose any more information than we need to.

3.6.1 Some properties of WI_1

WI_1 is intransitive and asymmetric in the general case. It is also non-monotonic with respect to additions to T.

1. **Intransitivity**

   $WI_1(\mathsf{f},\mathsf{g},\mathsf{T}) \wedge WI_1(\mathsf{g},\mathsf{h},\mathsf{T}) \not\Rightarrow WI_1(\mathsf{f},\mathsf{h},\mathsf{T})$

   For example, let f, g, and h stand for the following propositions.

   f : There is a blight on the tea crop in China.
   g : I am writing my thesis.
   h : The price of tea in China is affected.

   The general schema for generating intransitive claims is to take two causally related events and interpose an irrelevant one between them. This property of WI_1 makes it difficult to propagate irrelevance conclusions directly.

2. **Asymmetry**

   $WI_1(\mathsf{f},\mathsf{g},\mathsf{T}) \not\Rightarrow WI_1(\mathsf{g},\mathsf{f},\mathsf{T})$

   Let f and g stand for the following propositions.

   f : Joe sells his stock
   g : The stock market goes down
   T : Joe is a very small investor

It is true that Joe's selling his stock has a negligible effect on the market, but the crash of the market is very relevant to Joe's selling actions!

For the special case when f and g are independent (that is, they can be assigned truth values in T independent of each other) we can infer the irrelevance of g to f from the irrelevance of f to g.

3. **Non-monotonicity**

   Suppose we have an irrelevance claim that is true of a set of sentences. We will show that making consistent additions to that set does not necessarily preserve the irrelevance claim. We will use Example 1 defined earlier. We showed that $\neg WI_1(\mathsf{l}, \mathsf{t}, \mathsf{T}_2)$. Now, add t to T_2 to construct T_3. Applying Definition 15 again, we can establish that $WI_1(\mathsf{p}, \mathsf{q}, \mathsf{T}_3)$. This is an interesting kind of non-monotonicity: irrelevance claims are not preserved under deductive closure.

3.6.2 Towards a proof theory of WI_1

The irrelevance claim $WI_1(\mathsf{f}, \mathsf{g}, \mathsf{T})$ states that either f is not necessary for deriving g in T, or that there are alternative ways of deriving g that do not require f. The claim allows us to make statements about proofs of g in T without enumerating them. The assertional import of the claim (i.e. the inference it sanctions) is that it is *correct* with respect to preserving g, to simplify T to T' by the subset construction in Definition 15.

Example 3 *Let* $\mathsf{T} = \{\mathsf{p} \wedge \mathsf{q} \Rightarrow \mathsf{r},\ \mathsf{p} \wedge \mathsf{q} \Rightarrow \mathsf{s},\ \mathsf{s} \wedge \mathsf{t} \wedge \mathsf{u} \Rightarrow \mathsf{r},\ \mathsf{p} \Rightarrow \mathsf{t}, \mathsf{u}, \mathsf{p}, \mathsf{q}\}$

Notice that there are two ways of concluding r. The shorter proof uses p, q, and $\mathsf{p} \wedge \mathsf{q} \Rightarrow \mathsf{r}$. The longer one uses s as an intermediate conclusion. It is correct to assert in the meta-theory of T that $WI_1(\mathsf{s}, \mathsf{r}, \mathsf{T})$. If we act upon this claim, we would simplify T to the subset $\{\mathsf{p} \wedge \mathsf{q} \Rightarrow \mathsf{r},\ \mathsf{s} \wedge \mathsf{t} \wedge \mathsf{u} \Rightarrow \mathsf{r},\ \mathsf{p} \Rightarrow \mathsf{t}, \mathsf{u}, \mathsf{p}, \mathsf{q}\}$ and allow only the shorter proof of r to be concluded. The irrelevance claim is a control hint and the inference it sanctions shuts out undesired proofs of the goal. Suppose now that we want the longer proof of r to succeed. In this case, we would make the rule $\mathsf{p} \wedge \mathsf{q} \Rightarrow \mathsf{r}$ be irrelevant to r. The subset construction in Definition 15 does not achieve the omitting of the rule, which would ensure that the only proof of r remaining would be the one through s. To do this, we need the WI_2 notion here.

Definition 17 *$WI_2(f, g, T) \equiv T \models g$ and $\exists T'.\ T' \models g$ where $T' = T - \{f\}$.*

Example 4 *Let* $\mathsf{T} = \{\mathsf{p}, \mathsf{p} \Rightarrow \mathsf{q}, \mathsf{q}\}$

Is p irrelevant to q? By the subset definition, we can construct a subset T' of T, namely $\{\mathsf{p} \Rightarrow \mathsf{q}, \mathsf{q}\}$ that allows us to conclude q without committing ourselves on p. So p is indeed irrelevant to q. Let us consider the subsets of T and write down the irrelevance claims that would generate them.

$\{\mathsf{p}, \mathsf{p} \Rightarrow \mathsf{q}\}$:$WI_2(\mathsf{q}, \mathsf{q}, \mathsf{T})$

$\{\mathsf{p} \Rightarrow \mathsf{q}, \mathsf{q}\}$:$WI_1(\mathsf{p}, \mathsf{q}, \mathsf{T})$

$\{\mathsf{p}, \mathsf{q}\}$:$WI_2(p \Rightarrow \mathsf{q}, \mathsf{q}, \mathsf{T})$

This points to a limitation of WI_1, it cannot distinguish between stored and derived propositions. To generate the first subset above, we needed to remove q that was stored and allow the derived q to survive. WI_1 completely purges T of the proposition that is deemed irrelevant. So we need the weaker version WI_2 which simply removes the irrelevant proposition without removing those that derive it and those whose derivation depends on it. Intuitively, what WI_2 accomplishes is getting rid of f if it occurs *explicitly* in T, i.e., it removes 0-length proofs of f in T. This can be stated precisely as follows.

Theorem 5 *If* f *is a primitive fact in* T *(i.e. Support(*f,T*) = ∅), then WI_1(*f, g, T*) ≡ WI_2(*f,g, T*) and the* T' *that results from both constructions are the same.*

When f does not occur explicitly in T, then $WI_2(\mathsf{f}, \mathsf{g}, \mathsf{T})$ will be true vacuously. All that WI_2 can do is deem *cached* f's irrelevant to the computation of g. WI_1, on the other hand purges f completely from the formulation. So it removes cached f's as well as blocks all possible derivations of f in T.

3.6.3 An Axiomatization of WI_1

The domain of discourse of the logic includes propositional atoms, well-formed formulae (made from connectives), sets of wffs and proofs. The constants are taken from the set P of propositional atoms $p_1, p_2, \ldots, p_n$. The connectives are $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\equiv$. The well-formed formulaes are defined inductively. There are also sets of well-formed formulae $S_1, S_2, \ldots,$ and sequences of well-formed formulae (also called proofs) $P_1, P_2, \ldots$.

We now define some of the functions.

$Consequences$: wff $\times$ Set of wffs $\rightarrow$ Set of wffs
$Antecedents$: wff $\times$ Set of wffs $\rightarrow$ Set of wffs
$Intermediates$: wff $\times$ Set of wffs $\rightarrow$ Set of wffs
$Paths$: wff $\times$ Set of wffs $\rightarrow$ Set of Sequences of wffs

Informally, $Consequences$ finds the consequences of a wff in a set of wffs, both immediate and derived. $Antecedents$ finds the set of support for a wff in a set of wffs. $Intermediates$ finds those wffs in a set of wffs that use a given wff as an intermediate in their derivation. $Paths$ of a wff is a set of proofs for that wff.
These functions are defined inductively.

$\mathsf{x} \in Consequences(\mathsf{x}, \mathsf{T})$.
If $\mathsf{y} \in Consequences(\mathsf{x}, \mathsf{T}) \ \wedge \ \mathsf{y} \Rightarrow \mathsf{z} \in \mathsf{T}$ then $\mathsf{z} \in Consequences(\mathsf{x}, \mathsf{T})$.
$\mathsf{x} \in Antecedents(\mathsf{x}, \mathsf{T})$.
If $z \in Antecedents(\mathsf{x}, \mathsf{T}) \ \wedge \ \mathsf{y} \Rightarrow \mathsf{z} \in \mathsf{T}$ then $\mathsf{y} \in Antecedents(\mathsf{x}, \mathsf{T})$.
$Intermediates(\mathsf{h}, \mathsf{S}) = \{\mathsf{f}, \mathsf{g} \mid \mathsf{f} \Rightarrow *\mathsf{h} \in \mathsf{S} \wedge \mathsf{h} \Rightarrow *g \in \mathsf{S}\}$

$\Rightarrow^*$ stands for transitive closure under $\Rightarrow$.

Theorem 6 $WI_1(\mathsf{f}, \mathsf{g}, \mathsf{T})$ *is true exactly when* $\mathsf{f} \notin$ *Antecedents(*g, T*) or* f *is not in every element of Paths(*g, T*).*

The first condition corresponds to there being no directed path from f to g in the graph representation of T. The second condition is equivalent to saying that there is an alternate path to g that does not go via f.

3.6.4 Lemmas of WI_1

1. $WI_1(\mathsf{f}, \mathsf{g}, \mathsf{T}) \wedge WI_1(\mathsf{h}, \mathsf{g}, \mathsf{T}) \ \Rightarrow \ WI_1(\mathsf{f} \wedge \mathsf{h}, \mathsf{g}, \mathsf{T})$.

   Take an example: Let $\mathsf{T} = \{\mathsf{f}, \mathsf{h}, \mathsf{f} \Rightarrow \mathsf{g}, \mathsf{h} \Rightarrow \mathsf{g}\}$. $WI(\mathsf{f}, \mathsf{g}, \mathsf{T})$ because we can construct $\mathsf{T}_1 = \{\mathsf{h}, \mathsf{f} \Rightarrow \mathsf{g}, \mathsf{h} \Rightarrow \mathsf{g}\}$. Also, $WI_1(\mathsf{h}, \mathsf{g}, \mathsf{T})$ because we can construct $\mathsf{T}_2 = \{\mathsf{f}, \mathsf{f} \Rightarrow \mathsf{g}, \mathsf{h} \Rightarrow \mathsf{g}\}$. Now we ask whether $WI_1(\mathsf{f} \wedge \mathsf{h}, \mathsf{g}, \mathsf{T})$. Yes, because either T_1 or T_2 satisfies the requirement that neither $\mathsf{f} \wedge \mathsf{h}$ nor its negation is entailed in it.

2. $WI_1(\mathsf{f} \wedge \mathsf{h}, \mathsf{g}, \mathsf{T}) \ \Rightarrow \ WI_1(\mathsf{f}, \mathsf{g}, \mathsf{T}) \vee WI_1(\mathsf{h}, \mathsf{g}, \mathsf{T})$

To see why this is true: consider the following example. Let $\mathsf{T} = \{\, \mathsf{f}, \mathsf{h}, \mathsf{f} \Rightarrow \mathsf{g}, \mathsf{h} \Rightarrow \mathsf{g}\}$. There are two possible maximal subsets of T that can be constructed to show that $WI_1(\mathsf{f} \wedge \mathsf{h}, \mathsf{g}, \mathsf{T})$. One of them is T_1 from the previous example, and the other is T_2. If we only knew $WI_1(\mathsf{f} \wedge \mathsf{h}, \mathsf{g}, \mathsf{T})$, we couldn't tell whether f was irrelevant or h was irrelevant. So we can only conclude the disjunction of the individual WI_1 claims. This implication seems to be an iff one. Because if we can prove g without f, then we can prove g without f conjoined with h, which is stronger than f itself!

3. $WI_1(\mathsf{f} \vee \mathsf{h}, \mathsf{g}, \mathsf{T}) \Rightarrow WI_1(\mathsf{f}, \mathsf{g}, \mathsf{T}) \vee WI_1(\mathsf{h}, \mathsf{g}, \mathsf{T})$

   This can be proved in the following manner:

   $WI_1(\mathsf{f} \vee \mathsf{h}, \mathsf{g}, \mathsf{T}) \equiv WI_1(\neg(\mathsf{f} \vee \mathsf{h}), \mathsf{g}, \mathsf{T})$
   $\equiv WI_1(\neg\mathsf{f} \wedge \neg\mathsf{h}, \mathsf{g}, \mathsf{T})$
   $\Rightarrow WI_1(\neg\mathsf{f}, \mathsf{g}, \mathsf{T}) \vee WI_1(\neg\mathsf{h}, \mathsf{g}, \mathsf{T})$
   $\Rightarrow WI_1(\mathsf{f}, \mathsf{g}, \mathsf{T}) \vee WI_1(\mathsf{h}, \mathsf{g}, \mathsf{T})$

4. $WI_1(\mathsf{f}, \mathsf{g}_1 \wedge \mathsf{g}_2, \mathsf{T}) \Rightarrow WI_1(\mathsf{f}, \mathsf{g}_1, \mathsf{T}) \wedge WI_1(\mathsf{f}, \mathsf{g}_2, \mathsf{T})$

5. $\neg WI_1(\mathsf{f}, \mathsf{f}, \mathsf{T})$

   Straightforward consequence of the definition of WI_1.

6. $\mathsf{f}_1 \equiv \mathsf{f}_2 \Rightarrow [WI_1(\mathsf{f}_1, \mathsf{g}, \mathsf{T}) \Rightarrow WI_1(\mathsf{f}_2, \mathsf{g}, \mathsf{T})]$

   Follows directly from the definition of WI_1.

7. $\mathsf{g}_1 \equiv \mathsf{g}_2 \Rightarrow [WI_1(\mathsf{f}, \mathsf{g}_1, \mathsf{T}) \Rightarrow WI_1(\mathsf{f}, \mathsf{g}_2, \mathsf{T})]$

   Follows directly from the definition of WI_1.

8. $\mathsf{g} \in \mathsf{T} \wedge \mathsf{f} \not\equiv \mathsf{g} \Rightarrow WI_1(\mathsf{f}, \mathsf{g}, \mathsf{T})$

   If g is in T, then everything except itself is redundant to it.

9. $\mathsf{S} = \{\mathsf{p} \in \mathsf{T} | \ \mathsf{f} \Rightarrow \mathsf{p}\ \} \wedge WI_1(\bigwedge_{\mathsf{p} \in \mathsf{S}} \mathsf{p}, \mathsf{g}, \mathsf{T}) \Rightarrow WI_1(\mathsf{f}, \mathsf{g}, \mathsf{T})$ when $\mathsf{f} \not\Rightarrow \mathsf{g}$ in T

   If all facts derivable from f are redundant to g, then f itself is redundant to g.

10. $\mathsf{S} = \{\mathsf{p} \in \mathsf{T} \mid \mathsf{p} \Rightarrow \mathsf{g}\} \wedge WI_1(\mathsf{f}, \bigwedge_{\mathsf{p} \in \mathsf{S}} \mathsf{p}, \mathsf{T}) \Rightarrow WI_1(\mathsf{f}, \mathsf{g}, \mathsf{T})$ when $\mathsf{g} \not\Rightarrow \mathsf{f}$ in T

    Dual of previous lemma.

11. p$\in$T $\wedge$ Derives-Only(f,p,T) $\Rightarrow WI_1$(f,g,T) when T$\models$g

    If the only role of f is to derive p which already exists in T, then f is redundant for any goal other than itself.

Since detection of redundancy is undecidable, it is clear that we can never have a complete axiomatization of WI_1.

3.6.5 Properties of Proofs in the WI_1 calculus

Observation 4 *One benefit of proving redundancy of some facts in a theory, is that we can optimize space requirements by simply removing the redundant facts.*

A proof of redundancy using the lemmas of WI in the meta-theory of T captures an important property of the proofs of g in M, without exhaustively enumerating them. A problem solver that can prove this redundancy claim at the meta-level can prune a large class of inferences this way. Also if the redundancy statements were made available to the problem solver, it could compile it into the base level formulation by simply throwing away those facts that cause the redundancy. This leads to savings in space. To ensure that savings in time in *proving g* result from this pruning, we need to show that the proofs of g that fail as a result are derivationally more complex than the ones that are retained. This is captured in our definition of computational irrelevance below. In either case, the *deliberation time* for the problem solver will be reduced because the search space of proofs of g in T is reduced by the removal of redundancy.

Definition 18 CI_{max}*(*f,g,T*)* $\equiv$ T $\models$ g *and* $\exists$ T$'$. T$'$ $\models$ g *and* T$'$ $\not\models$ f*. where* T$'$ *is a maximal subset of* T *with this property and Cost(*T$'$ $\vdash$ g*)* $\leq$ *Cost(*T $\vdash$ g*).*

Observation 5 CI_{max}(f, g, T) $\Rightarrow WI_{max}$(f, g, T)*.*

The computational irrelevance notion folds in a utility measure; it orders logically irrelevant statements according to their computational impact. Contrast this with an approximate irrelevance claim that isolates logically relevant but computationally prohibitive distinctions made in a formulation.

3.6.6 Expressive power and limitations of WI_1

The WI_1 statements are not reducible; in the sense that every sentence in the extended language is not reducible to one in the pure propositional calculus. So WI_1 is more than a notational convenience, it is a genuine meta-theoretic notion. WI_1 formalizes reachability properties in a propositional dependency network of Horn clauses.

The subset construction is a very strong bias in the space of possible abstractions. The following is the simplest example of irrelevance that the subset definition does not cover.

Example 5 *Let* $\mathsf{T} = \{\neg \mathsf{f} \vee \mathsf{g}, \mathsf{f} \vee \mathsf{g}\}$.

f is irrelevant to g in T, because there are models of T that have f and g true, and f false and g true! Thus flipping the value of f in T does not cause that of g to change.

Theorem 7 *WI_1 provides an axiomatic account of a TMS style dependency analysis.*

1. Reachability analysis on inference graph
   The propositional logic of irrelevance WI_1 does a pure reachability analysis on the set of sentences in T. There are three cases of this reachability analysis.

   (a) $WI_2(\mathsf{p}, \mathsf{p}, \mathsf{T})$ iff $\mathsf{T} - \mathsf{p} \models \mathsf{p}$. p occurs explicitly in T and p is also derivable by other means in T. This happens when p is cached and the subset construction allows us to get rid of cached information. This is a construction that reduces space at the cost of time.

   (b) $WI_1(\mathsf{p}, \mathsf{q}, \mathsf{T})$ iff $\mathsf{T} \models \mathsf{q}$ and $\exists \mathsf{T}' \subset \mathsf{T}$ such that $\mathsf{T}' \models \mathsf{q} \wedge \mathsf{T}' \not\models \mathsf{p}$. This happens when there are two or more proofs of q in T, one that goes through p, and one that doesn't. The subset construction prunes out that part of the theory that allows the proof through p to succeed.

   (c) $Independent(\mathsf{p}, \mathsf{q}, \mathsf{T})$: the case where no proof of q uses p. Space savings only.

   These three cases take care of optimizations that save space and time without changing the ontology.

2. Collapsing nodes in inference graph
   The previous methods only short-circuit chains of reasoning within an inference graph while preserving its structure. This operation allows us to treat several nodes as a unit, effectively generating new objects that stand for equivalence classes of nodes.

3.6.7 Algorithms for computing WI_1

Here we present two basic methods for detecting whether f is irrelevant to g in the context of a theory T. We can break f into its constituents by the rules of the irrelevance calculus and then reduce all testing of irrelevance claims to proving atomic f's irrelevant.

- Break all ways of concluding f in T. Check if g is still deducible. One way of achieving this in a Horn clause database is to drop all clauses that have f occurring positively. If g is still deducible in the resulting database, we conclude that f is irrelevant to g.
- Compute the support set of g in T (i.e. find all ways of proving g in T). This is a set of justifications for g in T. Drop all justifications that contain f. If the support set empties after this, then f is not irrelevant to g. Otherwise it is.

The proof-theoretic correlate of these two procedures is immediately obvious. Method 1 foils all proofs of f at the last step (i.e. all premises of may be provable, but the deduction of f is not possible because the rule that concludes f from its premises is removed). So if a proof of g succeeds in the new set of clauses, f does not occur essentially in a proof of g. Method 2 computes all proofs of g and collects the disjunctive set of justifications. If f does not occur in all of them it again implies that f was not essential to proving g in T.

The first method above can be expressed as the following algorithm.

Algorithm 1 *1. Convert* T *to clausal form. 2. Remove all clauses that have a positive occurrence of* f *in them. Call the remaining set* T'. *3. Check if* T' *entails* g.

Since every way of concluding f is foiled in this construction, the resulting set T' does not entail f.

Example 6 $T = \{\mathsf{a}, \mathsf{a} \wedge \mathsf{b} \Rightarrow \mathsf{g}\}$

Is a irrelevant to g in this theory? No, because after we drop every positive occurrence (1 in this case) of a, g is not provable. In a sense what this procedure captures is the fact that if we prevent the concluding of a in T we would still be able to conclude g. Notice that this works well in Horn databases. But in non-Horn databases, this method fails.

Example 7 $\mathsf{T} = \{\mathsf{f} \vee \mathsf{g}, \neg\mathsf{f} \vee \mathsf{g}\}$. *We can resolve these two clauses and get* g. *The truth value assigned to* f *does not affect* g*;* g *remains true no matter what value we assign to* f.

Observation 6 *Algorithm 1 is sound with respect to the WI_1 definition.*

If the algorithm deems that f is irrelevant to g, then so would the definition of WI_1.

Observation 7 *Algorithm 1 is not complete with respect to the definition of WI_1.*

This algorithm generates *one* subset of T that does not entail f and checks if g is still entailed by that subset. There are many subsets of T that do not entail f, and if g is entailed by one that is not computed by Algorithm 1, then it will not be detected by it.

Observation 8 *Algorithm 1 does not compute maximal T' with the property required by the definition of WI_1.*

The clause $\neg h \vee f$ will be removed even if h is not derivable in T.

3.7 The first order case

For propositional logic, meta-level proofs of derivability offer no advantage over the base level proofs. Doing a step in the meta-level corresponds to doing an inference step at the base-level. For first-order logic, this is not the case and the real power of reasoning at the meta-level is demonstrated. The propositional definition of irrelevance scales up to the first order case as follows.

Definition 19 $\mathsf{f}(x, z)$ *is weakly irrelevant to* $\mathsf{g}(x, y)$ *modulo the set of sentences* T*, written as WI(*$\mathsf{f}(x, z), \mathsf{g}(x, y)$*,* T*), if we can construct the set* T' *such that* $\mathsf{T}' \subseteq \mathsf{T}$ *and* $[\forall xy.\ \mathsf{T} \models \mathsf{g}(x, y) \equiv \mathsf{T}' \models \mathsf{g}(x, y)$ *where* $[\forall z.\ \mathsf{T}' \not\models \mathsf{f}(x, z)]]$ *and* T' *is a maximal subset of* T *with this property.*

We show two examples of first-order irrelevance claims from the kinship example. As far as $\mathsf{SameFamily}$ is concerned, the distinction between immediate and non-immediate ancestry is irrelevant.

IC1: $\boxed{\forall xymn.\ \mathsf{Ancestor}(x, y) \in T \Rightarrow WI(\mathsf{Father}(x, y), \mathsf{SameFamily}(m, n), T)}$

Further, the identity of the common ancestor is also irrelevant; all that $\mathsf{SameFamily}$ seeks to establish is the existence of a common ancestor. This can be stated as follows:

IC2: $\boxed{\begin{array}{c}\forall xyzmn.\ \mathsf{Ancestor}(x,y) \in T \wedge \mathsf{Ancestor}(y,z) \in T \wedge \mathsf{Ancestor}(x,z) \in T \quad \Rightarrow \\ WI(\mathsf{Ancestor}(y,z), \mathsf{SameFamily}(\mathsf{m},\mathsf{n}), T)\end{array}}$

These irrelevance claims can be verified using the calculus of irrelevance presented below. We have built a meta-level irrelevance reasoner that can prove these claims given the initial encoding.

Verification of an irrelevance claim using the semantic definition of irrelevance takes time exponential in the size of T. To make the verification process tractable, we generate lemmas that satisfy the definition. These lemmas constitute the proof theory for WI. All variables below are assumed to be universally quantified; x and y are vectors of variables that are not disjoint.

1. $WI(\mathsf{f}_1(x) \wedge \mathsf{f}_2(x), \mathsf{g}(y), \mathsf{T}) \Rightarrow WI(\mathsf{f}_1(x), \mathsf{g}(y), \mathsf{T}) \vee WI(\mathsf{f}_2(x), \mathsf{g}(y), \mathsf{T})$

2. $WI(\mathsf{f}(x), \mathsf{g}_1(y) \wedge \mathsf{g}_2(y), \mathsf{T}) \Rightarrow WI(\mathsf{f}(x), \mathsf{g}_1(y), \mathsf{T}) \wedge WI(\mathsf{f}(x), \mathsf{g}_2(y), \mathsf{T})$

3. $\neg WI(\mathsf{f}(x), \mathsf{f}(x), \mathsf{T})$

4. $\mathsf{f}_1(x) \equiv \mathsf{f}_2(x) \Rightarrow [WI(\mathsf{f}_1(x), \mathsf{g}(y), \mathsf{T}) \Rightarrow WI(\mathsf{f}_2(x), \mathsf{g}(y), \mathsf{T})]$

5. $\mathsf{g}_1(y) \equiv \mathsf{g}_2(y) \Rightarrow [WI(\mathsf{f}(x), \mathsf{g}_1(y), \mathsf{T}) \Rightarrow WI(\mathsf{f}(x), \mathsf{g}_2(y), \mathsf{T})]$

6. $\mathsf{g}(y) \in \mathsf{T} \wedge \mathsf{f}(x) \not\equiv \mathsf{g}(y) \Rightarrow WI(\mathsf{f}(x), \mathsf{g}(y), \mathsf{T})$

7. $S = \{\mathsf{p}(x) \in \mathsf{T} \mid (\mathsf{f}(x) \Rightarrow \mathsf{p}(x)) \in \mathsf{T}\} \wedge WI(\bigwedge_{\mathsf{p}(x)\in S} \mathsf{p}(x), \mathsf{g}(y), \mathsf{T}) \Rightarrow WI(\mathsf{f}(x), \mathsf{g}(y), \mathsf{T})$

8. $S = \{\mathsf{p}(y) \in \mathsf{T} \mid (\mathsf{p}(y) \Rightarrow \mathsf{g}(y) \in \mathsf{T}\} \wedge WI(\mathsf{f}(x), \bigwedge_{\mathsf{p}(y)\in S} \mathsf{p}(y), \mathsf{T}) \Rightarrow WI(\mathsf{f}(x), \mathsf{g}(y), \mathsf{T})$

9. $\mathsf{q}(x) \in \mathsf{T} \wedge \ \mathit{Derives} - \mathit{Only}(\mathsf{f}(x), \mathsf{q}(x), \mathsf{T}) \Rightarrow WI(\mathsf{f}(x), \mathsf{g}(x), \mathsf{T}) \ \mathit{where}\ \mathsf{f}(x) \not\equiv \mathsf{g}(x)$

The lemmas 1-8 are straightforward extensions of the propositional lemmas. *Derives-Only* is a predicate that holds between fact schemas. *Derives-Only* ($\mathsf{f}(x)$,$\mathsf{q}(x)$,T) is true of an encoding T if the only rule that has f(x) on its left-hand-side concludes q(x). Since rules can be looked up in constant time by our problem solver, this check can be accomplished very quickly.

Theorem 8 *Lemmas 1 through 11 are sound with respect to the semantics presented in Definition 4.*

Theorem 9 *Lemmas 1 through 11 are incomplete with respect to the semantics presented in Definition 4.*

Proof: To complete the axiomatization of WI would require that we identify all the base cases for proving irrelevance (redundancy). Since detection of redundancy is semi-decidable, this task is impossible. □.

3.7.1 Proving Irrelevance Claims

Suppose we wish to prove **IC1**. We look for a lemma that matches the condition side of **IC1**. One lemma that does that is Lemma 9.

1. $\mathsf{q}(l) \in T \wedge Derives - Only(\mathsf{f}(l), \mathsf{q}(l), T) \Rightarrow WI(\mathsf{f}(l), \mathsf{g}(s), T) where \mathsf{f}(l) \not\equiv \mathsf{g}(s)$
   The unification process yields the following binding list $\{f = \mathsf{Father}, q = \mathsf{Ancestor}, g = \mathsf{SameFamily}, l = (x, y), s = (m, n)\}$. After instantiation, we obtain

2. $\mathsf{Ancestor}(x, y) \in T \ \wedge \ Derives - Only(\mathsf{Father}(x, y), \mathsf{Ancestor}(x, y), T) \Rightarrow$
   $WI(\mathsf{Father}(x, y), \mathsf{SameFamily}(m, n), T) where \mathsf{Father}(x, y) \not\equiv \mathsf{SameFamily}(m, n)$.

3. $Derives - Only(\mathsf{Father}(x, y), \mathsf{Ancestor}(x, y), T)$ follows from a lookup action.

4. $\mathsf{Father}(x, y) \not\equiv \mathsf{SameFamily}(m, n)$ follows from deduction over the definability structures introduced in Section 4.2.2
   After the resolution of 3 and 4 against 2, we have

5. $\mathsf{Ancestor}(x, y) \in T \Rightarrow WI(\mathsf{Father}(x, y), \mathsf{SameFamily}(m, n), T)$. □.

The claim above states that there are two ways of establishing any SameFamily query: those that use Ancestor(x,y) alone and those that use Father(x,y) as the terminating point. A problem solver that can prove this claim at the meta-level has the opportunity to choose between two classes of paths through the search space. If it prunes out the Father relation, all proofs of $SameFamily$ get shortened by 1 step and the search space is reduced because of the removal of nodes that result from the expansion using the $Father$ rule.

As in the propositional case, the pruning alters the base-level formulation so that this redundancy does not arise. Removal of irrelevant or redundant paths leads to savings both in space and time especially if the proofs of g that fail as a result of the irrelevance removal are the ones that are more complex. More importantly, the deliberation time for the problem solver is reduced because the search space is made smaller by the removal of redundant paths. This notion is made precise in the definition of computational irrelevance.

Definition 20 *$\mathsf{f}(x,z)$ is computationally irrelevant to $\mathsf{g}(x,y)$ given T, written as CI($\mathsf{f}(x,z)$,$\mathsf{g}(x,y)$,T), if it is the case that WI($\mathsf{f}(x,z)$,$\mathsf{g}(x,y)$,T), and the T' that results from the WI construction is such that AvgCost($\mathsf{T} \vdash \mathsf{g}(x,y)$) $\geq$ AvgCost($\mathsf{T}' \vdash \mathsf{g}(x,y)$).*

AvgCost is computed over a given distribution of the goal queries. **IC1** and **IC2** are computational irrelevance claims, because the construction of T''s dictated by them result in shorter chains of reasoning and smaller search spaces. These CI claims can be proven with the calculus of computational irrelevance, cost models for the problem solver and knowledge of the initial encoding.

Before we present a proof theory for CI, notice that **IC2** would be a valid WI claim if we replaced the conclusion by $WI(\mathsf{Ancestor}(x,z), \mathsf{SameFamily}(m,n), T)$. However, this claim does not cause reduction in proof height. In general, we have $\forall \mathsf{fgT}.\ CI(\mathsf{f},\mathsf{g},\mathsf{T}) \Rightarrow WI(\mathsf{f},\mathsf{g},\mathsf{T})$. While $WI(\mathsf{f},\mathsf{g},\mathsf{T})$ states that is it correct to remove f in the context of deducing g, the CI claim has the additional import that is worthwhile to remove f.

Proof theory of CI

We can construct the CI calculus by selecting and modifying exactly those lemmas from the WI set that actually reduce cost of proving g as defined by the cost model. This process is illustrated in the context of adapting Lemmas 8 and 11 to the CI calculus.

11. $\mathsf{q}(x) \in \mathsf{T} \wedge$ *Derives* $-$ *Only*$(\mathsf{f}(x), \mathsf{q}(x), \mathsf{T}) \Rightarrow WI(\mathsf{f}(x), \mathsf{g}(x), \mathsf{T})$ *where* $\mathsf{f}(x) \not\equiv \mathsf{g}(x)$

The two costs to compare are: height of the proof of $\mathsf{g}(x)$ if $\mathsf{f}(x)$ were present versus the height of the proof of $\mathsf{g}(x)$ if $\mathsf{q}(x)$ were ground instead. Since all proofs that use $\mathsf{f}(x)$ have to go through the expansion via the rule $\mathsf{f}(x) \Rightarrow \mathsf{q}(x)$, and since every ground f fact is replaced by exactly one q fact, the discarding of f meets both our time and space criteria. Lemma 9 is thus a CI lemma for our problem solver. CI lemmas tune the WI lemmas to the idiosyncracies of the given problem solver.

As another example, consider Lemma 8.

8. $\mathsf{g}(y) \in \mathsf{T} \wedge \mathsf{f}(x) \not\equiv \mathsf{g}(y) \Rightarrow WI(\mathsf{f}(x), \mathsf{g}(y), \mathsf{T})$

The only cost to consider here is that of storing $\mathsf{g}(y)$. If $|g|$ is less than the space bounds allotted to our problem solver, we would include this as a CI lemma. Notice that our reformulation cost measures do not include the cost of computing the relation g. The time and space bounds are on the end product of the reformulation and not on the computation of the end product. This is because we amortize the cost of reformulation over all future uses of the new formulation with better computational properties.

3.7.2 Information needed to prove irrelevance

The information needed to demonstrate that a certain class of facts or distinctions are logically irrelevant to a specified class of goals includes

1. Information about the given encoding

   We need to know what relations are represented explicitly/implicitly, the number of facts in the encoding that match a given form, information about the completeness of the database for a given relation, as well as information about the stability of the encoding for a given relation. As an example of the latter, consider the situation in the kinship example where Father facts can be retracted later. In this case, we cannot destroy the distinction between Father and Ancestor facts, because doing so will prevent us from integrating new Father facts as the appropriate Ancestor facts. Definedness graphs introduced in Chapter 5 maintain dependencies between various parts of the encoding in a very efficient manner.

2. Information about the conceptualization

   Properties of the objects and relations that are encoding independent, e.g., associativity, commutativity, and symmetry of relations.

3. Information about the goals

   We take the goals directly into account in our formulation of irrelevance. The frequency and distribution of queries is an important factor in the determination of what information can be ignored.

3.8 Related Areas

A formal study of irrelevance reveals that there is interesting substructure in the space of useful irrelevance inferences. There is a hierarchy of irrelevance logics that is similar to

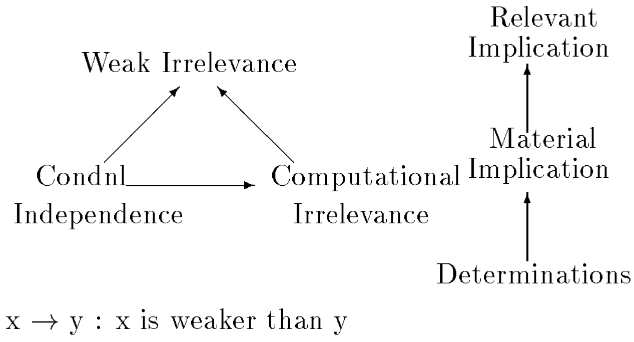


Figure 3.2: Logics of Relevance and Irrelevance

the hierarchy of logics that capture the notion of relevance [AB75, DR87]. See Figure 3.2. These logics are not exact duals of each other; many aspects of the relationship between them remain to be investigated. The statement that f is *relevant* to g in T conveys the information that knowing f in T restricts the space of possibilities for g. The statement that f is *irrelevant* to g in M indicates that the value of g is insensitive to the value of f, in that even if f were changed in T, g would not. In the missionaries and cannibals puzzle, the names of the missionaries are irrelevant to the scheduling of the boat trips. This means that even if the names were changed[3], the solution (which does not refer to the missionaries by name) would not. This irrelevance statement gives us the justification to modify the formulation in which missionaries are named, to the more abstract one that uses the cardinality of the set of missionaries.

The two notions of relevance and irrelevance are complementary; one expresses a dependence between two facts and the other captures a one-sided lack of dependence (g being not dependent on f). However, the normal use of an irrelevance statement is to modify T while preserving g, the normal use of a relevance statement is to infer restrictions on g given f: thus the inferences they sanction are not duals.

[3] as long as they remain distinct from each other!

Chapter 4

Generating Reformulations

4.1 Generation = Discovery + reduction

This chapter describes how to *use* irrelevance claims to generate reformulations. The previous chapter outlined the descriptional import of irrelevance claims – here we discuss their assertional import, i.e. the inferences that they sanction. We present a framework for the generation of abstraction reformulations: first, the discovery of appropriate irrelevance claims in the meta-theory of a formulation, and then the reduction of the formulation by inferences that minimize irrelevant distinctions. This method of generating abstractions by minimizing distinctions irrelevant to a given class of goals can be treated as a first-principles account of abstraction. We discuss the logical character of the reduction and the discovery processes and analyze their computational complexity. We state the irrelevance principle which advises minimization of distinctions modulo a goal and show that it underlies the construction of a large class of abstractions.

4.2 The Irrelevance Principle

Irrelevance claims, both logical and computational, identify irrelevancies and redundancies in the formulation of a problem. For instance, in the kinship example that we have used so far, the first irrelevance claim

IC1: $\boxed{\forall xymn.\ \mathsf{Ancestor}(x,y) \in T \Rightarrow WI(\mathsf{Father}(x,y), \mathsf{SameFamily}(m,n), T)}$

expressed the fact that the distinction between immediate and non-immediate ancestry represented by the relations **Father** and **Ancestor** respectively, is irrelevant to the **Same-Family** relation. A reduction inference uses this irrelevance claim to modify the formulation so that it no longer makes this irrelevant distinction.

One modification that a reduction inference can perform is to relabel all Father facts as Ancestor facts, effectively erasing the distinction between the relations they denote. This inference is sanctioned by the irrelevance principle whose informal statement is: removing distinctions irrelevant to a goal schema leads to the construction of computationally tractable theories for that class of goals. Note that this is a local optimisation principle that justifies hill-climbing in the space of conceptualizations toward reducing distinctions. The global irrelevance principle states that the best reconceptualization makes the fewest distinctions consistent with the correctness and goodness constraints given. That is, the optimal formulation is the one that doesn't make any more distinctions than is logically or computationally necessary. We will call both these principles *representational irrelevance principles* since their domain of discourse is conceptualizations. We can conceive of a similar set of principles that talk about computations performed by an encoding. The local version of such a principle advises hill-climbing in the space of encodings toward fewer computations that is consistent with the correctness and goodness constraints. The global version of this principle states that the best re-encoding is one that doesn't do any more computation than is necessary to achieve the goal within the given space constraints. We shall call these two principles the local and global *computational irrelevance principles* respectively. The representational and computational principles are akin to Snell's law for computational systems: they propose taking the *path of least resistance* in the space of representations and computations.

The representational irrelevance principle is a variation of Quine's principle of indiscernibility of identicals [Qui63]. It is re-echoed in Harman's [Har86] principle of clutter avoidance and Lenat's notion of cognitive economy [LHRK79]. Whereas the representational irrelevance principle outlaws making unnecessary distinctions, the computational irrelevance principle outlaws unneeded computation. To apply the computational irrelevance principle we need to specify an encoding and a problem solver to determine the computations that are performed. The calculus of computational irrelevance is a declarative way of specifying the distinctions in a formulation that give rise to wasted computation in a problem solver.

We can state the representational and computational irrelevance principles formally.

To do that we shall introduce two definitions.

Definition 21 *A conceptualization* $\mathcal{C}_1 = (\mathcal{O}_1, \mathcal{F}_1, \mathcal{R}_1)$ *makes fewer distinctions than the conceptualization* $\mathcal{C}_2 = (\mathcal{O}_2, \mathcal{F}_2, \mathcal{R}_2)$ *with respect to the goal relation* $\mathbf{G}$ *, written as Makes-fewer-distinctions(*$\mathcal{C}_1, \mathcal{C}_2, \mathbf{G}$*), if Definable(*$\mathbf{G}$,$\mathcal{C}_1$*)* $\wedge$ *Definable(*$\mathbf{G}$,$\mathcal{C}_2$*)* $\wedge$ *Definable-C(*$\mathcal{C}_1, \mathcal{C}_2$*).*

That is, a conceptualization $\mathcal{C}_1$ makes fewer distinctions than $\mathcal{C}_2$ in the service of $\mathbf{G}$ if $\mathbf{G}$ is definable in both, and if the objects, functions and relations of $\mathcal{C}_1$ are constructible from those of $\mathcal{C}_2$.
We wish to find the formulation which makes the fewest distinctions that can *compute* G most efficiently. To do that, we need to define a metric on encodings that determines if one encoding performs less computation than another in the service of the same goal.

Definition 22 *Encoding* E_1 *of the conceptualization* $\mathcal{C}_1$ *computes* $\mathbf{G}$ *more efficiently than encoding* E_2 *of* $\mathcal{C}_2$ *if* $\mathbf{G}$ *is definable in* $\mathcal{C}_1$ *and* $\mathcal{C}_2$*, and if the cost of proving the encoding* G_1 *of* $\mathbf{G}$ *in* E_1 *is less than that of proving the encoding* G_2 *of* $\mathbf{G}$ *in* E_2 *with respect to the given cost metric* C *on the problem solver PS. We then say that Makes-fewer-computations(*$E_1, E_2, \mathbf{G}, C, PS$*).*

The local representational irrelevance principle (LRIP) can be stated as:
LRIP: $Makes\text{-}fewer\text{-}distinctions(\mathcal{C}_i, \mathcal{C}_j, \mathbf{G})$
$\Rightarrow Better\text{-}Conceptualization(\mathcal{C}_i, \mathcal{C}_j, \mathbf{G})$

A conceptualization is better than another if it can express $\mathbf{G}$ while making fewer distinctions. The global representational irrelevance principle states that the best conceptualization for $\mathbf{G}$ is such that

GRIP: For a given goal $\mathbf{G}$, select $\mathcal{C}_i$ such that $\neg\exists \mathcal{C}_j.\ Better\text{-}Conceptualization(\mathcal{C}_i, \mathcal{C}_j, \mathbf{G})$.

Clearly, the $\mathcal{C}_i$ that satisfies this constraint is one that contains $\mathbf{G}$ alone! This is because in a world where there are no computational constraints, the best formulation for a problem is one that directly contains the answers: a giant lookup table for $\mathbf{G}$. To take computational constraints into account, we modify the LRIP so that not only does a better conceptualization make fewer distinctions, but that it permits an encoding that makes fewer computations according to some cost metric C on a problem solver PS.

The modified local representational irrelevance principle (LCRIP) states that

LCRIP:

$$\begin{aligned} \forall \mathcal{C}_{\mathrm{i}}\ \mathcal{C}_{\mathrm{j}} \quad & \textit{Makes-fewer-distinctions}(\mathcal{C}_i, \mathcal{C}_j, \mathbf{G}) \\ \wedge\ & \exists PS.\ \exists C.\ \exists E_k.\ (\textit{Encoding}(E_k, \mathcal{C}_i) \\ \wedge\ & \forall E_l.\ \textit{Encoding}(E_l, \mathcal{C}_j)\ \wedge\ \textit{Makes-fewer-computations}(E_k, E_l, \mathbf{G}, C, PS)) \\ & \Rightarrow \textit{Better-Conceptualization}(\mathcal{C}_i, \mathcal{C}_j, \mathbf{G}) \end{aligned}$$

The local computational irrelevance principle (LCIP) states that

LCIP:$\textit{Makes-fewer-computations}(E_i, E_j, \mathbf{G}, C, PS) \Rightarrow \textit{Better-Encoding}(E_i, E_j, \mathbf{G}, C, PS)$

An encoding is better than another if it performs fewer computations in the service of **G**. The global computational irrelevance principle states that the best encoding for **G** with respect to a given problem solver and a cost metric is such that

GCIP: For a given goal **G**, a problem solver *PS* and cost metric C, select E_i such that $\neg\exists E_j.\ \textit{Better-Encoding}(E_i, E_j, \mathbf{G}, C, PS)$

If there are multiple E_i's with this property, we choose one which encodes a conceptualization with the fewest distinctions. This intuition is expressed in the modified LCIP below.

LRCIP:

$$\begin{aligned} \forall E_i\ E_j\ \exists C.\ \exists PS. \quad & \textit{Makes-fewer-computations}(E_i, E_j, \mathbf{G}, C, PS) \\ \wedge\ & \exists \mathcal{C}_k.\ (\textit{Encoding}(E_i, \mathcal{C}_k) \\ \wedge\ & \forall \mathcal{C}_l.\ \textit{Encoding}(E_j, \mathcal{C}_l)\ \wedge\ \textit{Makes-fewer-distinctions}(\mathcal{C}_i, \mathcal{C}_j, \mathbf{G})) \\ & \Rightarrow \textit{Better-Encoding}(E_i, E_j, \mathbf{G}, C, PS) \end{aligned}$$

How should the search for a good reconceptualization proceed? The irrelevance principles provide natural gradients in the space of all conceptualizations. One approach is to start with the given conceptualization and incrementally *remove* distinctions to arrive at one that meets the given correctness and goodness constraints. This is the approach used in the thesis. A complementary approach is to begin with a conceptualization that has **G** alone and incrementally *add* distinctions to meet the computational constraints. The problem of determining which distinction to *add* is significantly harder than the problem of determining which to remove, because the number of distinctions that could potentially be added far exceeds those that can be removed. However, when this condition is violated, the second approach to traversing the space of conceptualizations is the preferable one.

The irrelevance principles as stated are normative. We can determine whether a particular conceptualization shift or an encoding shift respects the irrelevance principle. Thus, the conceptual shift in the kinship example that required introducing the new relation **FoundingFather** and eliminating **Father** for solving the goal relation **SameFamily**, obeys the LCRIP. This is because the new conceptualization is definable in terms of the old one, and thus makes fewer distinctions. Also, it can be encoded to meet the given time and space constraints. The compression of the transitivity chains in the encoding of the same example by introducing the new predicate symbol FoundingFather is an encoding shift that is better for any non-caching, deductive problem-solver under any reasonable cost metric (proof heights, size of the search space). This allows us to show that the encoding shift in the kinship example follows the LCIP. Demonstrating that a particular conceptualization or encoding satisfies GRIP or GCIP, is extremely difficult because of the enumeration over all possible conceptualizations and encodings for **G**. In this thesis we focus entirely on satisfying the local versions of the two principles.

The representational and computational irrelevance principles are related; for some problem solvers and for some classes of encodings, minimizing distinctions while preserving the goal actually leads to minimizing computation. In fact, CI claims, introduced in Chapter 3 capture exactly those cases where removing information (minimizing distinctions) leads to removal of unneeded or redundant computation.

4.3 Reduction Inferences

What is the role of the irrelevance principle in generating reformulations? We propose reduction inferences that operate on encodings of conceptualizations and that use irrelevance claims. The task of the reduction inference is to implement the minimization of distinctions suggested by the LRIP or the minimization of computation as suggested by the LCIP. Since the irrelevance claims themselves are in the realm of encodings (e.g., **IC1**), a reduction inference can operationally interpret a claim of the form $c \Rightarrow WI(f, g, T)$ to be: enriching a formulation to make the condition on the left hand side of the claim true, allows the removal of the facts in it that are specified as irrelevant. This is the situation

$$\frac{\text{Theory + Meta-theoretic Irrelevance Claim IC}}{\text{New-Theory where IC is not true of the meta-theory of New-Theory}}\ \text{Reduction Inference}$$

Figure 4.1: The form of a reduction inference

depicted in Figure 2.12. The move at the meta-theoretic level is toward a formulation in which none of the irrelevance claims can be derived; i.e., a formulation that doesn't make any of the distinctions deemed irrelevant to the class of goals.

This section describes the mechanics of modifying a formulation using irrelevance claims. Figure 4.1 describes the form of a reduction inference. We introduce the relation *Reduce* that takes a formulation and an irrelevance claim and produces a new formulation in whose meta-theory the irrelevance claim does not hold.

Definition 23 *An irrelevance claim IC of the form $c \Rightarrow$ WI(f,g,T) holds in the meta-theory of T, if we can apply the T' construction of definition 15 on T augmented to make c hold.*

If the T' constructed is equal to T[1], we say that IC does not hold in the meta-theory of T. The construction in Definition 15 does not specify T' uniquely, so we constrain the choice by requiring it to be the largest subset of T that satisfies the definition. $Reduce(IC, WI(f,g,T)) = T'$ where T' is the largest subset of T that satisfies Definition 15. Unfortunately, this still does not fix T' uniquely. Consider the following example.

Example 8 *Let $T=\{a, b, a \Rightarrow g, b \Rightarrow g\}$. We can prove that WI($a \wedge b, g, T$). If we reduce T by this claim, we have two candidate T''s. ${T_1}' = \{a, a \Rightarrow g, b \Rightarrow g\}$ and ${T_2}' = \{b, b \Rightarrow g, a \Rightarrow g\}$.*

The class of T's and WI claims for which a unique T' can be found is generalized by the following theorem.[2]

[1] It violates the strict subset requirement.

[2] When the T''s are not unique, the reducer makes a choice which may later be retracted. This issue is discussed in the implementation of a reduction system. The search space for reduction is a function of the cardinality of the possible T''s at each point.

Theorem 10 *Reduce(T,IC) produces a unique T' for sets T that have no disjunctions and for claims IC= WI(f,g,T) that have non-conjunctive f's.*

Proof: If T has disjunctions, it might allow multiple derivations of g that don't involve f, and this would cause multiple T''s to be generated. Since $\mathrm{WI}(f_1 \wedge f_2, g, T) \Rightarrow \mathrm{WI}(f_1, g, T) \vee \mathrm{WI}(f_2, g, T)$, conjunctive f's automatically create multiple maximal subsets of T that derive g without entailing f. $\Box$.

We would like to let reduction inferences change the formulation *minimally* to remove irrelevant distinctions. This policy specifies a preference among the formulations that result by revising a given one to eliminate a distinction: it conservatively picks a theory that is "closest" to the original theory. The reason that this is a computationally sensible policy to adopt is that it minimizes the work done by the reduction process itself.

In order to reduce a formulation with respect to an irrelevance claim, we apply the construction of Definition 15 until the irrelevance claim no longer holds in the meta-theory of the new formulation. When this happens, the T' returned by the construction is identical to T.

Theorem 11 *The reduction process with respect to a claim IC terminates when Reduce(T_n,IC)=T_n.*

Proof: When this condition obtains, the irrelevance claim IC is no longer true in the meta-theory and reduction with respect to it has to terminate. $\Box$.

The iterative application of *Reduce* implements the meta-theoretic movement to the empty set of irrelevance claims depicted in Figure 2.12. The following theorems also hold.

Theorem 12 *Applying Reduce iteratively with respect to a given WI claim until a fix point of the theory is reached, produces a new conceptualization that is better as defined by the LRIP.*

Proof: Every reduction by a WI claim removes some fact or a class of facts from a theory. The theory gets weaker since its deductive closure decreases, and the number of models it has increases. Each model of the new theory can be created by dropping elements from a model of the old theory. We can treat conceptualizations as special models which include the Herbrand base and the named functions and relations on that base. Thus the new conceptualization can be defined in terms of the old one. Hence the new conceptualization makes fewer distinctions and is a better conceptualization in the LRIP sense. $\Box$.

Theorem 13 *Applying Reduce iteratively with respect to a given CI claim until a fix point is reached, produces a new encoding that is better as defined by the LCIP.*

Proof: By construction, reduction by a CI claim guarantees that the removal of a fact or a class of facts from the given encoding causes the goal to be solved faster in the sense defined by the cost-metric C for a given problem solver PS. So the new encoding performs fewer computations in the service of the goal schema, and by the LCIP it is a better encoding. □.

Depending on whether the reduction process operates directly on the present formulation or on a description of it, we have two modes of reduction: extensional and intensional.

4.3.1 Extensional Reduction

Extensional reduction performs direct surgery on the formulation as dictated by the irrelevance claims and generates a new formulation that obeys both the goodness and the correctness constraints. We give a non-procedural account of the *ExtReduce* action that achieves the reduction. *ExtReduce* is a special case of *Reduce* that has the following I-O behaviour.

Inputs:
- A set of sentences T
- A set I of meta-theoretical claims of irrelevance in T with respect to goal schema **g**

Output:
- A new set of sentences T' in whose meta-theory no element of I can be derived.

This form of reduction is called extensional reduction because it works with the sentences in T and with ground instances of the irrelevance claims. For instance, if we instantiate **IC1** with bindings $\{x{=}\mathrm{A}, y{=}\mathrm{B}\}$ and call it **IC1**(A,B) and then compute the value of *ExtReduce*(T,**IC1**(A,B)): we will find it equal to be $T \cup$ {Ancestor(A,B)} − {(Father(A,B)}. This constitutes one step of the reduction process that relabels Father links as Ancestor links throughout the tree. We can use **IC1** again to reduce T'. The process terminates when all Father facts are labelled as Ancestor facts. Then **IC1** no longer holds in the meta-theory and the construction in Definition 15 makes no more changes to the input theory.

Theorem 14 *The extensional reduction process terminates when ExtReduce(T_n,IC)=T_n.*

Theorem 15 *If the irrelevance claim IC expressed in prenex normal form has n quantifiers, then the number of extensional reduction steps that need to be performed is bounded by the product of the domain sizes of the quantifiers.*

Proof: By construction. *ExtReduce* has to obtain *all* ground instances of the claim before it can perform the reduction, so in the worst case we get the above upper bound. $\Box$.

4.3.2 Intensional Reduction

The input-output specification of intensional reduction *IntReduce* is as follows:

Inputs:	•	A description of a set of sentences T
	•	A set I of meta-theoretical claims of irrelevance in T with respect to the goal schema **g**
Output:	•	A new description of a set of sentences T' in whose meta-theory no element of I can be derived.

One way to conceptualize intensional reduction is to think of it as limit reasoning [Wel86] over extensional reduction. Intensional reduction takes a *description* of the formulation as well as the irrelevance claims and produces a *description* of the reduced formulation that results after the irrelevance minimization process terminates. We present an example to clarify this idea.

Suppose that all the Father facts in Figure 1.3 have already been relabelled as Ancestor facts. Now we want to minimize the formulation using **IC2**. While the extensional reduction process laboriously rewires the family tree by applying *ExtReduce* over and over, intensional reduction attempts to find a description of the facts left at the *end* of the extensional reduction. A description of the Ancestor facts at the start is:

$$\forall u.\ T \models \mathsf{Ancestor}(root, u).$$

We wish to find the description of T' such that Reduce(T',**IC2**) $= T'$. $\mathsf{Ancestor}(root, u)$ is shorthand for the formula $\mathsf{Ancestor}(x, u) \wedge \neg\exists m.\ \mathsf{Ancestor}(m, x)$.
The reasoning proceeds in two steps.

1. **IC2** preserves connectedness to the root.
   $\forall u.\ T \models \mathsf{Ancestor}(root, u) \Rightarrow Reduce(T, \mathbf{IC2}) \models \mathsf{Ancestor}(root, u)$

2. When the reduction terminates **IC2** can no longer be applied.

   By examining the preconditions of **IC2**, we see that the required condition is that all $\mathsf{Ancestor}$ links be one step long; i.e., $\neg\exists xyz.\ \mathsf{Ancestor}(x, y) \wedge \mathsf{Ancestor}(y, z)$.

Putting 1 and 2 together, we obtain the fact that all $\mathsf{Ancestor}$ links are connected to the root and that they are all one step long. That is, the remaining $\mathsf{Ancestor}$ facts satisfy: $\mathsf{Ancestor}(root, y)$ which is an abbreviation for $\mathsf{Ancestor}(x, y) \wedge \neg\exists m.\mathsf{Ancestor}(m, x)$ which is the required definition of the $\mathsf{Founding\text{-}Father}$ relation!

Mathematical induction was required here because of the recursion on $\mathsf{Ancestor}$. Note that the description of the initial formulation was used directly as the invariant in the induction proof. The termination condition was obtained by negating the left hand side of **IC2**.

4.3.3 Properties of Reduction

The form of the base level theory

How sensitive are the irrelevance claims to the actual encoding? Suppose we had instead the following encoding of the kinship problem.

$\mathsf{Father}(x, y) \Rightarrow \mathsf{Ancestor}(x, y)$

$\mathsf{Father}(x, y) \wedge \mathsf{Ancestor}(y, z) \Rightarrow \mathsf{Ancestor}(x, z)$

$\mathsf{Ancestor}(z, x) \wedge \mathsf{Ancestor}(z, y) \Rightarrow \mathsf{SameFamily}(x, y)$

The irrelevance claims are the same; the reduction procedure has to be sensitive to the particular encoding. We now show how *Reduce* transforms the formulation using **IC1**. It is clear that all ground Father facts will get replaced by ground $\mathsf{Ancestor}$ facts. The inference sanctioned by **IC1** is the replacement of the schema $\mathsf{Father}(x, y)$ by the schema $\mathsf{Ancestor}(x, y)$ in the theory. We do the replacement throughout the formulation and obtain the rules $\mathsf{Ancestor}(x, y) \Rightarrow \mathsf{Ancestor}(x, z) \wedge \mathsf{Ancestor}(z, y)$

Domain of discourse of the irrelevance claim

The complexity of the reduction inferences is a function of the domain of discourse of the irrelevance claims. For the kinship example, the irrelevance claims were on particular classes of facts, already in the formulation. The complexity arose from attempting to find an intensional characterization of the reduction. For the Fibonacci transformation, the irrelevance claims refer to repeated computations, and the reduction process has to determine the changes to the formulation that ensure that those computations are not redone.

Order of Reduction

When there is more than one irrelevance claim, there is a non-deterministic choice of the order of reduction. One natural question to ask is whether the order of reduction of irrelevance claims makes a difference to the end result; i.e. whether reductions satisfy the Church-Rosser property. The answer is that in the cases where it is meaningful to reduce in any order, reductions do not necessarily obey the Church Rosser property. When the claims are independent, then the order is irrelevant. In our example, if we start with a database containing Father facts alone, the analysis of the preconditions of **IC1** and **IC2** show that **IC1** needs to be applied before **IC2**. So there is only one order of reduction. Suppose for instance that the effect of an irrelevance claim **IC3** is to take the transitive reduction of the database and the effect of claim **IC4** is to take the transitive closure of a database, it is clear that the order of reduction makes a significant difference to the end result.

Termination of Reduction

How can we know that the reduction process terminates? We can reason that the reduction process terminates on the kinship example by using the fact that the databases are finite and that the relations are acyclic. Suppose we attempt to reduce a graph denoting paths between adjacent rooms as in Figure 4.3 with **IC2**. The reduction process will not terminate: every possible cycle on three nodes will be generated. Standard methods of termination reasoning[Man74] can be used to determine non-termination in this case. However, the fact that termination reduces to the halting problem poses fundamental limits on inferring termination.

4.3.4 Another Formulation of Irrelevance Minimization

Here we formulate the problem of minimizing irrelevant distinctions in a formulation in terms of minimizing computational entities like proofs and search spaces. The predicate $Proof(p, g, T)$ represents the fact that p is a proof of the goal g within the encoding T. If g is a goal that is defined by a wff with a prefix existential quantifier [3], then we can define a relation $Witness$ that extracts the value of the existentially quantified variable

[3](e.g., SameFamily(x, y) is defined to be true if there *exists* a z such that z is the ancestor of both x and y)

in a proof of g.

For the kinship example, we have the following property of the proofs of SameFamily.

$\forall x \ y \ Proof(p, \mathsf{SameFamily}(x, y), E_1) \ \wedge \ Witness(p, z) \ \wedge \ E_1 \models \mathsf{Ancestor}(z, z') \Rightarrow \exists p' \ z' \ Proof(p', \mathsf{SameFamily}(x, y), E_1) \ \wedge \ Witness(p', z')$

We can implement the LCIP by minimizing the number of proofs that a goal schema has in a given encoding. The proliferation of proofs in E_1 is caused by the the transitivity of the Ancestor relation, and the statement above expresses it. If we allow a proof of SameFamily(x, y) that uses a witness z, we will have to accept all other proofs of the same fact that cite the ancestors of z as witnesses. To minimize the number of proofs for each SameFamily fact, we simply pick the "highest" proof and modify the formulation to eliminate the smaller proofs that cite lower ancestors as witnesses. This transformation in proof space translates to introducing the FoundingFather relation in the formulation space. In general, this minimization process introduces new terms that stand for macro-objects in the formulation space and macro-actions in the search space.

Whereas traditional circumscription minimizes the extents of predicates and objects in a theory to construct minimal models of the theory, we minimize the objects, and extents of predicates in order to minimize computational entities like space, intermediate computation etc., that result in solving a goal schema within that theory.

4.4 The discovery of irrelevance claims

4.4.1 By being told

This is the easiest way of acquiring them. As we have shown earlier, incorporating irrelevance claims into a theory is a non-trivial endeavour. In general, the stronger the impact of the reduction, the harder it is to discover them automatically. The irrelevance claim however, can be verified by the irrelevance reasoner.

4.4.2 By derivation in the meta-theory

We use the lemmas of WI and CI to suggest possible irrelevance claims. These lemmas have the form: *Condition on formulation* $\Rightarrow$ *WI(fact-schema,goal-schema,formulation)*. We "backward chain" on these lemmas by setting up the goal of proving a certain *fact-schema* irrelevant to the given *goal-schema*. We illustrate this by an example:

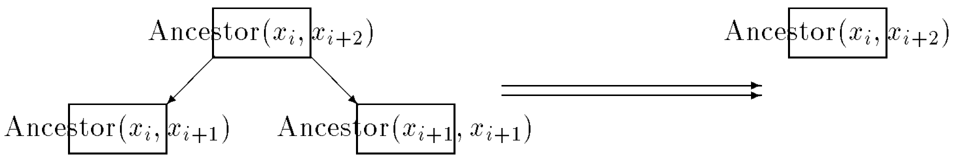


Figure 4.2: Replacing a subtree by a node in the proof of SameFamily

Take WI lemma 11.

1. $\mathsf{q}(l) \in \mathsf{T} \wedge \ Derives\!-\!Only(\mathsf{f}(l), \mathsf{q}(l), \mathsf{T}) \Rightarrow WI(\mathsf{f}(l), \mathsf{g}(s), \mathsf{T}) \ where \ \mathsf{f}(l) \not\equiv \mathsf{g}(s)$

and resolve it against the fact that

2. $Derives\!-\!Only(\mathsf{Father}(x,y), \mathsf{Ancestor}(x,y))$ which is true of the formulation.

This resolution generates the following binding list $\{\mathsf{f} = \mathsf{Father}, l = (x,y), s = (m,n)\}$. Given that the goal is $\mathsf{SameFamily}(a,b)$, we now verify that $\mathsf{Father}(x,y) \not\equiv \mathsf{SameFamily}(a,b)$ which is easy given the definedness graphs that are introduced in Chapter 5 and obtain

3. $\mathsf{Ancestor}(x,y) \in T \Rightarrow WI(\mathsf{Father}(x,y), \mathsf{SameFamily}(m,n), T)$ which is the required claim.

Goal-directed derivation of irrelevance claims in the meta-theory is only as good as the lemmas in the calculus of WI and CI. The ultimate source of irrelevance claims is in empirical analyses of computational structures as well as statistical correlations between attributes in the world.

4.4.3 By empirical analyses of proof and search spaces

We can use the declarative specification of the problem solver to construct the symbolic computation trace for the goal g. In our example, all SameFamily proofs end in Father facts. The fringe of all these proof trees can be moved up one step closer to the root without increasing the width of the tree, if the proofs are made to terminate in the corresponding Ancestor facts. This transformation of the proof tree can be accomplished by relabelling the Father tree as the Ancestor tree and by getting rid of the Father rule. The proof tree transformation is a schema for reconfiguring proofs and the irrelevance claim expresses what the corresponding formulation transformation should be.

The second irrelevance claim results from another general proof tree transformation that attempts to reduce the height of a proof. However, this requires that we examine a portion of the proof tree shown in Figure 4.2. We essentially try to replace a section of the tree by a single node by the elimination of intermediate variables. In this example, there are two ways of achieving it: getting rid of the $\mathsf{Ancestor}(x_i, x_{i+1})$ branch, or the $\mathsf{Ancestor}(x_{i+1}, x_{i+2})$ branch. The correctness constraint filters out the incorrect branch to prune, and we thus derive IC2. The proof transformation above is a very general schema for reconfiguring proofs because it also underlies the construction of Thevenin equivalents of an analog circuit.

For the particular problem solver we are considering, short proofs are preferable to long ones. Therefore, transformations that influence the height of the proof like the ones above are very useful for reformulations that improve computational efficiency with respect to this problem solver. In order to cover more general classes of proof transformations and search space transformations (e..g., eliminate dead ends), a language for expressing and manipulating proof transformations is needed.

4.5 Reformulations Justified by the Irrelevance Principle

The kinship example of this thesis is a reformulation of an equivalence relation as a partition; a standard transformation taught to every computer scientist. The method of irrelevance minimization explains how this transformation can be derived from more general considerations.

This section demonstrates that a large number of abstractions and optimisation methods can be analyzed using the theory of irrelevance. For each example, we write the meta-theoretic irrelevance claims that justify the abstraction and demonstrate the reduction methods used to minimize irrelevant distinctions. The examples we consider are: variants of the kinship example, Thevenin equivalents and tail-recursion optimizations (e.g. Fibonacci).

4.5.1 Variants of the Kinship Example

Notice that if the goal relation in the kinship example had been the **CommonAncestor** relation, which is true of a triple (x, y, z) just when z is the common ancestor of x and y, then no information in the initial formulation can be proven to be irrelevant to this goal. So the compression of ancestor links cannot be done. This demonstrates that the abstraction method is sensitive to the goals.

Now we will show how some more variations of the kinship example can be analyzed using irrelevance. Suppose we have the following formulation of connectivity between

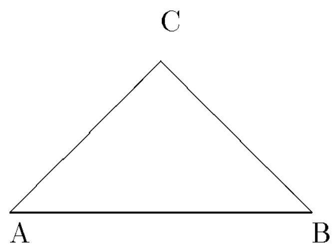


Figure 4.3: A case in which extensional reduction does not terminate

rooms. The relation **DConn** is a list of pairs of rooms that are directly connected. **Conn** is the transitive closure of **Dconn**. The goal relation is **Samecluster**: two rooms belong to the same cluster if they are connected to a common room. This problem is an ablation of the Founding Fathers example: the **Father** relation is not symmetric, whereas **Dconn** is. An encoding for the SameCluster problem is given below.

$\mathsf{DConn}(x, y) \Rightarrow \mathsf{Conn}(x, y)$

$\mathsf{Conn}(x, y) \wedge \mathsf{Conn}(y, z) \Rightarrow \mathsf{Conn}(x, z)$

$\mathsf{Conn}(z, x) \wedge \mathsf{Conn}(z, y) \Rightarrow \mathsf{SameCluster}(x, y)$

$\mathsf{Conn}(x, y) \Rightarrow \mathsf{Conn}(y, x)$

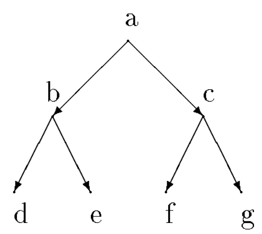
The irrelevance claims that are true in the meta-theory of this encoding are:

IC3: $\boxed{\forall xymn.\ \mathsf{Conn}(x, y) \in T \Rightarrow WI(\mathsf{DConn}(x, y), \mathsf{SameCluster}(m, n), T)}$

IC4: $\boxed{\begin{array}{c}\forall xyzmn.\ \mathsf{Conn}(x, y) \in T \wedge \mathsf{Conn}(y, z) \in T \wedge \mathsf{Conn}(x, z) \in T \quad \Rightarrow \\ WI(\mathsf{Conn}(y, z), \mathsf{SameCluster}(m, n), T)\end{array}}$

IC5: $\boxed{\begin{array}{c}\forall xyzmn.\ \mathsf{Conn}(x, y) \in T \wedge \mathsf{Conn}(y, z) \in T \wedge \mathsf{Conn}(x, z) \in T \quad \Rightarrow \\ WI(\mathsf{Conn}(x, y), \mathsf{SameCluster}(m, n), T)\end{array}}$

If we applied extensional reduction to the encoding with the claims **IC3** through **IC5**, the reduction will not terminate. The condition on the left-hand side of **IC4** and **IC5** will never be false in the original encoding or its reductions. The lack of directionality in the DConn facts makes the character of this problem very different from the SameFamily problem. This is evident in the following statement in the space of proofs. E is the above



Father(a,b)
Father(a,c)
Father(b,d)
Father(b,e)
Father(c,f)
Father(c,g)
SameFamily(x,x)
Father(x,y) $\Rightarrow$ SameFamily(x,y)
Father(y,x) $\Rightarrow$ SameFamily(x,y)
SameFamily(z_1,z_2) $\wedge$ Father(z_1,x) $\wedge$ Father(z_2,y) $\Rightarrow$ SameFamily(x,y)

Figure 4.4: The encoding E_5 of $\mathcal{C}_1$

encoding.

$\forall x\ y\ Proof(p, \mathsf{SameCluster}(x, y), E) \wedge Witness(p, z) \wedge$
$E \models \mathsf{SameCluster}(z, z') \Rightarrow \exists p'\ z'\ Proof(p', \mathsf{SameCluster}(x, y), E) \wedge Witness(p', z')$

A graph-theoretic interpretation of the reduction is informative. Minimizing the heights of all proofs of SameCluster(x, y) requires that we minimize the sum of the length of paths from node x to y for every x and y in the graph. This value is minimized when all nodes point to a common node. We can do this either by picking a particular node in the graph, or by introducing a new object to stand for a cluster representative. In the SameFamily problem, the fact that Father was directed provided us with a canonical representative of the equivalence relation SameFamily.

What if the kinship problem had been formulated entirely in terms of the **Father** relation? An encoding that achieves this is shown in Figure 4.4.

The irrelevance claims **IC1** and **IC2** do not hold directly in the formulation; they hold counterfactually. This is because, this formulation has no explicit redundancy. The computation of the **SameFamily** relation is done by finding the *least* common ancestor of two people. In order to reformulate this encoding to meet the computational constraint

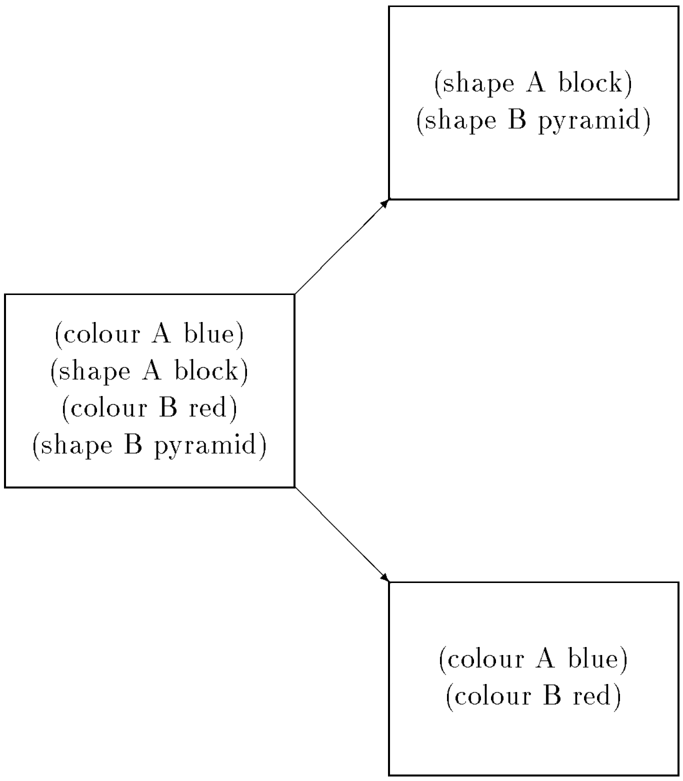


Figure 4.5: Compiling out factored knowledge bases

of solving for Samefamily queries in constant time with O(n) overhead in space (n=the number of people), we have to introduce the redundant relation Ancestor. If we interpreted **IC1**, counterfactually, we would relabel the Father relation as the Ancestor relation. If **IC2** were treated the same way, we would add the Ancestor links on the left-hand side of the claim, in order to render Ancestor(y, z) irrelevant. This would have the effect, atleast extensionally, of generating the maximal ancestor links. The intensional reduction inferences are non-trivial because the system will have to infer the transitivity of Ancestor that is hidden in the claim **IC2**.

4.5.2 Irrelevance and the factoring of knowledge bases

Definition 24 *A theory T is factorable if and only if $\exists T_1 T_2 \ldots T_n$ such that*

- $\cup_{i=1}^{n} T_i = T$
- $T - T_i \not\models T_i$

Figure 4.5 shows a set of sentences T about the colors and shapes of some blocks. Since the relations **Shape** and **Color** are independent in the world, we can factor T into T_1 and T_2 as shown in the figure. We can derive this factoring as a reduction via an irrelevance claim that expresses the independence of Shape and Color.

$\forall x, y, m\ WI(Shape(x, y), Color(x, n), T) \wedge WI(Color(x, n), Shape(x, y), T)$

This irrelevance claim is true in the meta-theory of T. To reduce T to make the irrelevance claim false, we factor T into T_1 in whose meta-theory the first conjunct of the above irrelevance claim is false, and T_2 in whose meta-theory the second conjunct of the claim is false. This reduction inference did not introduce any new terms. It is a pure efficiency transformation – the new theories are indexed in a manner that allows a forward chaining problem solver to answer questions about the color of blocks (resp. about shapes) without deriving irrelevant intermediate facts about their shapes (resp. about colors).

It is useful to look at this transformation from an irrelevance perspective. Reorganizing a set of facts in a theory in a non-random way to make the solution of a given set of queries more efficient takes work. There is work involved in

1. Acquiring the appropriate claims of irrelevance. In this case, these claims about the independence of relations.

2. Reducing the formulation to eliminate irrelevance. In this case, the independence claims could be incorporated into the theory by factoring.

The factored formulation as well as the original formulation have the same information content at the base level; which is reflected by the fact that they are equivalent at the model-theoretic level. However, at the meta-level, different irrelevance claims hold of them; and this is reflected in the differences in their structural properties. The following theorem demonstrates the fact that the re-indexing of a theory accomplished by factoring has computational utility.

Theorem 16 *Factoring of theories can be justified by independence claims of the form WI(p,q,T) $\wedge$ WI(q,p,T). Factoring T into T_1 and T_2 that satisfies Definition 24 achieves the reduction of irrelevance.*

To make the claims in this section more rigorous, we need to set up a more detailed model of the problem solver and provide the distribution of queries with respect to which the irrelevance or independence claims are determined. Then we can demonstrate that the re-encoded theory performs fewer computations (or database lookups) to solve the given set of queries. We can then propose quantitative measures of ease of re-organization or re-indexing of a theory in terms of the complexity of discovering the necessary irrelevance claims and reducing the theory with respect to them.

4.5.3 Applications of LCIP

The computational irrelevance principle states that an encoding that does fewer computations to achieve a certain goal is better. This is a minimization principle that governs the way we design representations and optimise computations on those representations. If we can parameterize a computation we can find the values at which work is minimized given some cost model of our problem solver. There are three basic ways in which we can eliminate computation in the service of a goal.

1. Eliminate computations done by an encoding that do not contribute to the goal. An example is the following
   $\mathsf{Parent}(x, y) \Rightarrow \mathsf{Ancestor}(x, y)$
   $\mathsf{Parent}(x, z) \wedge \mathsf{Ancestor}(z, y) \Rightarrow \mathsf{Ancestor}(x, y)$

   Suppose the query is: Ancestor(John,y). Suppose further that we have a forward chaining problem solver that computes all ancestor facts from the given parent facts and then projects out the ancestors of John. Rewriting the encoding in the manner given below ensures that only the relevant ancestors are computed. It involves introducing a new predicate magic-ancestor of arity one. The method of rewriting is the magic set method [BR87].

   $\mathsf{magic} - \mathsf{ancestor}(John)$
   $\mathsf{magic} - \mathsf{ancestor}(x) \wedge \mathsf{Parent}(x, y) \Rightarrow \mathsf{magic} - \mathsf{ancestor}(y)$
   $\mathsf{magic} - \mathsf{ancestor}(x) \wedge \mathsf{Parent}(x, z) \wedge \mathsf{magic} - \mathsf{ancestor}(z) \wedge \mathsf{Ancestor}(z, y) \Rightarrow \mathsf{Ancestor}(x, y)$
   It can be shown that the magic set method produces encodings that perform fewer unneeded computations to solve a goal schema. Reformulations generated by this method thus implement the intent of the LCIP.

   This essentially precomputes the rule set so that the parts of the search space that do not contribute to the goal are removed at compile time. Some relations are pruned as a result. There is computational

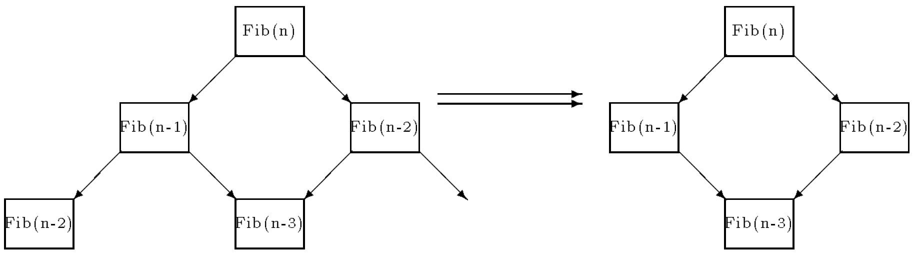


Figure 4.6: Restructuring a part of the Fibonacci computation

savings obtained; because useless computation is avoided; also in a system that has to choose between alternate inference steps valuable decision time at the meta-level has been saved.

2. Eliminating repeated computation

   This is done by reifying that computation; introducing a new object that stands for the value of that computation and looking up the value as opposed to storing it. There are two cases to analyse here: repeating computation while solving a single goal as in the computation of Fibonacci or Factorial, and repeating computation over several instances of solving for a set of related goals, an instance of which is common subexpression elimination. The standard formulation of Fibonacci

   $\mathsf{Fib}(0, 1)$.

   $\mathsf{Fib}(1, 1)$.

   $\mathsf{Fib}(n-1, m_1) \wedge \mathsf{Fib}(n-2, m_2) \wedge m = m_1 + m_2 \Rightarrow \mathsf{Fib}(n, m)$

   run on a depth-first backward chainer produces the computation trace shown in Figure 4.6. We can show in the meta-theory of this encoding that repeated calls to $\mathsf{Fib}(m, -)$ in the computation of $\mathsf{Fib}(n, -)$ where $m < n$ occurs Fib(n-m) times (except for $\mathsf{Fib}(0, -)$ which occurs Fib(n-2) times). We wish to devise an encoding that does not perform repeated computation: in effect, an encoding that generates a trace as in Figure 4.6 when executed by a depth-first backward chainer. The transformation at the level of proof trees is the conversion of a computation tree into a directed acyclic graph (DAG) that has common nodes merged. The reformulation problem is to find the corresponding change in encoding that causes this transformation in proof space. This situation is depicted in Figure 4.7.

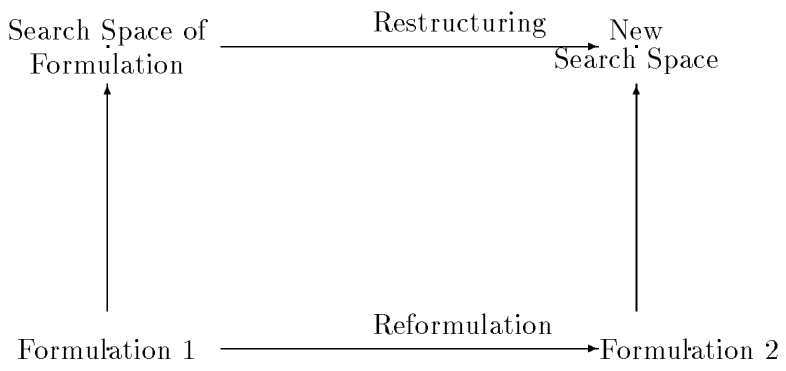


Figure 4.7: Redesigning formulations by restructuring search spaces

Given

- A problem solver PS that takes an encoding E of a problem and a goal schema G and generates a proof P for it. PS(E,G) = P
- Given a standard transformation of P to P' that avoids repeated computation.

Find an encoding E' such that PS(E',G) = P'.

A solution to this requires that we able to correlate changes in proof space with changes in formulation space. One approach simply maintains a table of proof space changes and the encoding changes that implement them. For instance, to merge identical nodes in computation space, we simply create a new object to store that value. In the Fibonacci case, adding an extra argument to the Fib function achieves this:

$\mathsf{Fib}(n, a, b) \Leftarrow \text{if} n = 0 \text{then} a \text{else} \mathsf{Fib}(n-1, a+b, a)$

To find the fibonacci of n, we use the query $\mathsf{Fib}(n, 1, 1)$.

Another approach requires us to have the inverse function PS^{-1} that maps proofs back to encodings. Then the corresponding encoding change can be *deduced* from the change in proofs. Unfortunately, this idea cannot be implemented at the present because we lack good descriptions of problem-solvers. When these descriptions become available, this method will be a general-purpose scheme for deducing new encodings driven by computational constraints.

The changes in proof space can be captured indirectly by the following irrelevance claim (from Chapter 3).

IC3: $\boxed{\forall xn.\, Computed(x) \Rightarrow WIA(Compute(x), Factorial(n), T)}$

This states that if a value has already been computed, then the *action* of computing it is irrelevant with respect to solving a goal. This advises the interpreter to store a value as soon as it is computed. For the Fibonacci function, the elimination of unnecessary computation can be achieved either by changing the formulation while keeping the problem solver (or interpreter) fixed, or by changing the interpreter itself while keeping the encoding fixed. This latter possibility is implemented by the reduction method that compiles WIA claims.

3. Eliminate computations that contribute to higher accuracy that we don't care about. This is the basis for constructing approximations. For instance, the computations that result from having the base-emitter capacitor in the hybrid-π model of a transistor, can be eliminated under low frequency conditions since these computations only contribute to the higher-order terms in the gain. The two variants of the computational irrelevance principle introduced above can be weakened by weakening the definition of logically irrelevant computation: if a computation contributes only to the higher order bits of accuracy we will call it irrelevant. The elimination of repeated computation uses an exact match to determine when two computations are identical. We can relax the criterion to get approximate matches and this would lead to the elimination of almost equivalent computations.

4.5.4 General Abstractions

A taxonomy of irrelevance abstractions can be constructed based on the reduction framework described in this chapter.

- Answer a subset of queries, the same answer needed.
  For this class, the irrelevance claims are modulo the subset of queries that need to be preserved. Reduction by these claims leads to the generation of new defined relations. The resulting formulation is a specialization of the given formulation tuned to answering those queries really efficiently. The kinship examples in this chapter are examples of this.

- Same queries, abstract answer suffices

  The abstract answers defined an equivalence class of solutions. Irrelevance claims propagate equivalence on solution space into the formulation. When a certain property is deemed irrelevant then all objects distinguishable by that property become indistinguishable. Abstraction to remove properties has the side-effect of creating equivalence classes of objects. Examples are the missionaries and cannibals problem as well as Abstrips.

- Same queries need to be solved at the same level of detail

  Find wasted computation and state it as irrelevance claims. Reducing irrelevance usually involves introducing a new object or relation to store computation that was getting repeated before. Examples are Fibonacci, factorial, macrops and compiler optimisations.

Chapter 5

Techniques for Automating Irrelevance Analyses

This chapter proposes reformulation techniques that are specializations of the meta-level irrelevance minimization described in the previous chapters. The two processes that need to be compiled to provide effective automation of reformulation are

1. Discovery of irrelevance claims.

2. Reduction of formulations by the irrelevance claims.

We analyze various types of irrelevance reformulations and describe the special properties of the irrelevance claims that allow for the development of tractable algorithms for the discovery and reduction of irrelevance.

5.1 Intermediate variable elimination

The key idea explored here is optimising computations by eliminating intermediate computation. We recognize intermediate computation and determine if it is irrelevant. These are stated as computational irrelevance claims in the meta-theory. We then rewrite the computation so that only endpoints are preserved. This is implemented by reduction inferences that shortcircuit intermediate computation.

A simple example where reasoning chains can be compressed without violating correctness constraints is the following.

Example 9 *The set* $T_1 = \{p(x,y) \wedge q(y,z) \wedge r(z,w) \Rightarrow s(w)\}$ *can be reformulated as follows by introducing a new intermediate relation that eliminates the thread through the variable* y.
$T_2 = \{t(x,z) \wedge r(z,w) \Rightarrow s(w)\}$ *where the articulation theory* A *between* T_1 *and* T_2 *is* $\{p(x,y) \wedge q(y,z) \Rightarrow t(x,z)\}$.

The correctness proof for this reformulation is trivial. In the first place, $T_1, A \models T_2$ and $T_2, A \models T_1$. Also, it is the case that $\forall w.\ T_1 \models s(w) \equiv T_2 \models s(w)$ for the same extensional database (same extensions for given predicates p, q and r). The difference in the two programs surfaces in the efficiency proof. In the proof space, we reduce the branching factor on the $s(w)$ node by 1 by introducing the pre-computation of the join of p and q. Essentially, we prevent recomputation of $p(x,y) \wedge q(y,z)$ over various instantiations of the goal $s(w)$. y is a thread variable that does not occur in the final query. The role of z is also to act as an intermediary in the computation of $s(w)$. One idea is to deem all intermediates irrelevant – this will have the effect of minimizing intermediate variable threading, by precomputing joins of relations and materializing those joins. If only the endpoints matter, then store them and throw the intermediate stuff away. This is the idea behind chunking and macrops.

The order of thread variable elimination determines the size of the intermediate relations introduced. The thread variables that we eliminate depend on the queries to be preserved, the sizes of relation, and the sizes of intermediate relations created. In the simplest cases, the thread variables are immediately apparent in the encoding. Almost all of the dynamic programming examples have this property. However, recognition of irrelevant intermediate computation is far from easy. The recognition that the threading through Ancestor could be eliminated in the kinship example is a non-trivial one.

We now perform an analysis using irrelevance for thread varibale elimination. There are two possible analyses.

Analysis 1

$\forall x \in T \Rightarrow WIA(Compute(x), g, T)$

Captures the fact that if a fact x is in the database, the action of computing it is irrelevant. It assumes that x is an eternal fact (does not change with time). Compute(x) reifies the search tree for x rooted at x. The search tree for x is expanded using the rules in T.

The semantics of WIA is: even if the action Compute(x) is not performed, g will be solvable in the T as long as x is in it. It prescribes that the action Compute(x) not be performed. Instead to obtain the value of x, a lookup in the database will be performed.

Analysis 2

The irrelevance claim for Example 9 is

$t(x, z) \in T \Rightarrow CI(p(x, y), g(m, n), T) \wedge CI(q(y, z), g(m, n), T)$

A gross step in the computation (from x to z), renders its component steps computationally irrelevant. In fact, the irrelevance claim in Analysis 1 is a special case of this; since the two options: looking up a value and computing it are compared and the gross step of looking up a stored value (bypassing the computation) renders the steps that compute the value irrelevant.

The reduction method for this class of abstractions, simply adds the new relation on the LHS of the irrelevance claim, and removes the component relations from which it is derived.

The complexity of doing thread variable elimination in cases where the threads are recursive is best illustrated by the kinship example. Here are some comparisons between the kinship example and the p-q example. The main similarities are that in both cases

- a new relation is introduced that is definable entirely in terms of the existing relations.
- a certain class of queries is preserved across the transformation.
- the set of objects is not changed.
- the new relations FoundingFather and t are good islands of pre-computation.

The main differences are

- the new relation **FoundingFather** is a subset of the given **Ancestor** relation. So a problem solver that simply caches Ancestor facts would ultimately obtain the links that make up the new relation. t is not a subset of an existing relation. The effect of introducing the relation symbol t cannot be duplicated by caching ground atomic literals.
- t is a simple macro-operator, whereas FoundingFather is not.

- introducing the FoundingFather relation symbol eliminates redundancy in the space of proofs (now every SameFamily fact has exactly one proof), whereas the introduction of t does not.

The method of eliminating thread variables can be applied to any state-space problem formulated as follows.

$Legal(s_1, s) \wedge Reachable(s, s_2) \Rightarrow Reachable(s, s_2)$

This states that state s_2 is reachable from s_1 if there is a legal move from s_1 to an intermediate state s from where s_2 is reachable. Now we need to minimize the *Reachable* relation while preserving paths of interest to prevent unnecessary intermediate computation.

5.1.1 Relations to previous work

Comparison to partial evaluation: the general techniques of partial evaluation involve propagation of constants and unfolding calls by their definitions. Generally, they do not make short cuts in the computation sequence or change program control to eliminate redundant computation: both of these are done by the methods to automate irrelevance reformulations.

Comparison with macrops: justifications for short-circuiting steps in search space or proof space is implicit in the creation of macrops. The method of irrelevance minimization first introduces redundant macro steps and then eliminates the constituent micro steps. The macrops question is: given a state space what are the best short circuits to construct for solving a given class of goals efficiently? The abstraction problem with irrelevance minimization is: what steps need to be added and what can be thrown away in the service of a given class of goals. Note that a system using macrops needs to learn control knowledge to make the problem solver choose appropriate short circuits. The irrelevance minimization method analytically computes this knowledge and compiles it into the base theory.

5.2 Irrelevance analysis of abstraction in circuits

The aim is to do an irrelevance analysis of some classic abstractions in digital and analog circuits. As an example from analog circuits, we use the theory of irrelevance to derive the concept of Thevenin equivalents. This analysis helps to clarify the computational utility

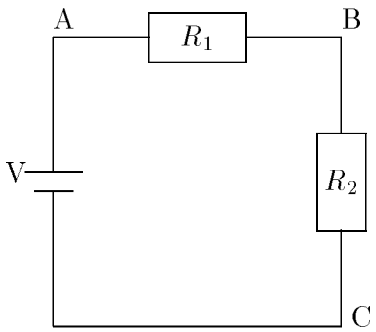


Figure 5.1: A simple resistive circuit

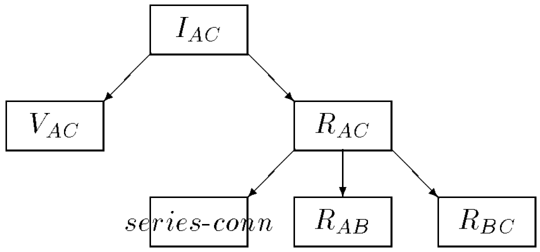


Figure 5.2: Proof of the goal I_{AC}

of forming Thevenin equivalents of circuits in terms of the search space transformation it entails. As an example from digital circuits, we construct the structural abstraction of a digital device [Sin86]. This reformulation effectively hides detail about intermediate computation in the circuit.

5.2.1 Thevenin Equivalents

In the simple resistive network of Figure 5.1, the goal is to compute the current I_{AC} through the voltage source V. The standard method of solution uses Ohm's and Kirchoff's laws. The proof tree that solves this goal is in Figure 5.2.

To meet our computational constraint of solving for I_{AC} in constant time with a constant overhead in space, we decide to shorten proofs of I_{AC}. Terminating the proof at height 1 satisfies the constraint on time. To make sure that all future instances of proofs

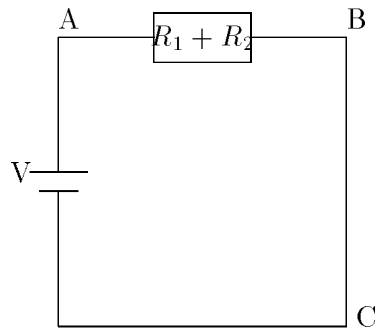


Figure 5.3: The Thevenin equivalent reformulation

of I_{AC} are 1 step long, we propagate the proof tree transformation into the formulation by replacing the R_1,R_2 series combination of resistors by the equivalent R_{AC}. The introduction of this new object achieves the effect of reducing the height of the proof for this and future instances of proofs of I_{AC} on this circuit. This transformation is equivalent to replacing an entire subtree in the computation by a single node and reifying the value of the node in a new object.

We can give an account of the introduction of a new object in terms of reduction of irrelevance. The value of the voltage at the point B is irrelevant to the computation of the current through the circuit. So we can weaken the initial formulation to make the value of the voltage at B underivable without affecting the value of I_{AC}. The irrelevance claim is:

$WI(V_{AB}, I_{AC}, T) \wedge WI(V_{BC}, I_{AC}, T)$

Reducing the formulation to make these claims underivable in the meta-theory of the new formulation requires introducing the compound object $R_{AC} = R_{AB} + R_{BC}$ and rewriting the formulation as in Figure 5.3.

The form of the reduction inference is:

Given:

- the detailed circuit description
- Ohm's and Kirchoff's laws
- Description of goal schema

- the irrelevance claims with respect to the goal schema

Find a new abstract description that does not compute the values specified as irrelevant by the given claims.

5.2.2 Full Adders

Consider a digital device specified as follows: $IMP(a,x,y,g) \equiv P_1(a,x,y,x) \wedge P_2(y,y,g)$. To solve for values of g given values of a, we have to compute the internal values x and y. However, as far as the output value g is concerned, the value at the intermediate points is irrelevant and can be abstracted. This requires weakening the theory of the device by introducing a new predicate that existentially quantifies the internal signals. $\mathsf{New} - \mathsf{Pred}(a,g) \equiv \exists x, y\ IMP(a,x,y,g)$. Notice that the new predicate introduced hides information about the internal structure of the device and preserves its i-o behaviour. Singh [Sin86] calls these abstractions structural abstractions. At the level of proof trees, this amounts to shortening the proofs to height 0 by essentially precomputing the i-o behaviour as a table in the relation New-Pred.

In the case of the full adder, shown in Figure 2.1, we can derive the structural abstraction that eliminates details of the actual connections between the various gates, by specifying the values at the internal points, viz., d, e and f irrelevant. To reduce the theory with respect to these irrelevance claims, we need to weaken it to make the values at these points underivable in the new theory. The introduction of the new predicate $Full - Adder(a,b,c,sum,carry) \equiv sum = xor(a,b,c) \wedge carry = ab + bc + ca$ and removing the original description of the full adder, makes the values at d,e and f no longer deducible. There are two major steps in the abstraction: identifying which points in the circuit to treat as internal points (since sum and $carry$ are the outputs we are interested in, and a, b and c are the given inputs, all other nodes in this circuit become internal nodes by elimination), and derivation of the expressions for sum and $carry$ purely in terms of the givens: which is an algebraic rewrite problem.

5.3 The Irrelevance theorem prover

5.3.1 The propositional version

We now present an implementation of a propositional logic of irrelevance that allows us to calculate irrelevancies in time linear in the size of the dependency network. The basic idea is the following: have a dependency network at a stable configuration with truth values attached such that each node has values that make it true. We then perturb the network, by changing some value from true to false, or false to true. Then we propogate the effects of this through the network with local propagation rules that calculate the partial discrete derivatives of these propositions.

5.3.2 The Implementation of irrelevance detection by hardware

The basic idea is to convert a set of sentences into a combinatorial circuit so it can answer questions of the form "Is f irrelevant to g?" in the following manner. We perturb the value of f in the circuit (if f is an internal point, we have to modify the input appropriately, to achieve the change a f) and propagate the consequences down to g.

We can do this only if the theory is finite, there are no cyclic dependencies in the circuit and there are no recursive rules.

The propagation rules are (f,h are inputs to a gate, g is the output)

1. AND: if h = 1 then $f(0 \rightarrow 1) \Rightarrow g(0 \rightarrow 1)$
2. AND: if h = 1 then $f(1 \rightarrow 0) \Rightarrow g(1 \rightarrow 0)$
3. OR : if h = 0 then $f(0 \rightarrow 1) \Rightarrow g(0 \rightarrow 1)$
4. OR : if h = 0 then $f(1 \rightarrow 0) \Rightarrow g(1 \rightarrow 0)$
5. NOT: if $f(1 \rightarrow 0) \Rightarrow g(0 \rightarrow 1)$
6. NOT: if $f(0 \rightarrow 1) \Rightarrow g(1 \rightarrow 0)$

More complicated f's can be reduced to atomic ones and the above propagation methods applied.

5.4 The irrelevance reduction system

5.4.1 Extensional Reduction

The extensional reduction algorithm takes the following inputs:

1. A database of facts and rules that describes a problem
2. The goal
3. The irrelevance claims for that goal

and produces a new database of facts and rules that preserves answers to the goal. The irrelevance claims are no longer true of the new database.

The algorithm is:

```
Algorithm Reduce
Inputs: Theory T to be reformulated
        Irrelevance claims of the form Condition => CI(f,g,T)
Output: New theory where irrelevance claims are no longer trivially true

 Temp-Theory = Emptyset.
           T = Input Theory                    ; initialization
 Repeat until T and Temp-Theory are the same
  Temp-Theory = T.
  Find an irrelevance claim where f unifies with a fact in T.
  If it is a conditional claim then
      Find augmentation action to make condition true
      Perform augmentation action.
  Find revision action to make f underivable.
  Perform revision action.
 End Repeat
```

Augmentation actions indexed by the form of the condition to be achieved are coded as condition-action rules. Revision lemmas that prescribe particular delete actions are the assertional component of the irrelevance calculus. We illustrate one step of reduction by **IC1** on a database. Suppose $\mathrm{T} = \{\mathsf{Father}(A, B), \mathsf{Father}(B, C), \mathsf{Father}(x, y) \Rightarrow \mathsf{Ancestor}(x, y)\}$.
IC1: $\forall xymn.\ \mathsf{Ancestor}(x, y) \in T \Rightarrow CI(\mathsf{Father}(x, y), \mathsf{SameFamily}(m, n), T)$.

Name	:	*IC1 (&x &y)*
Condition	:	*Father(&x &y) in T*
Addlist	:	*Ancestor(&x &y)*
Deletelist	:	*Father(&x &y)*

Figure 5.4: The STRIPS operator compiled from Irrelevance Claim 1

If we follow the steps of the algorithm above after the appropriate initializations;

1. We select IC1 with bindings $\{x = A, y = B\}$.

2. We now have the goal of augmenting the theory to make Ancestor$(A, B) \in T$ true. We use the fact that an $\in$-of condition can be made true by adding the fact named in the condition to the theory. We then perform the action of adding Ancestor(A, B) to T.

3. Now the goal is to revise the augmented theory to make it not entail Father(A, B). We use the fact that Father is a primitive relation (it is a source node in the definability map) and the fact that revising a theory to make a primitive fact underivable simply requires removing it. We then perform the action of removing Father(A, B).

Note that the reasoning steps required to establish the augmentation and revision methods are repeated for every instantiation of Father(x, y) in the database. We compile out this reasoning, by performing them once and for all at the start of the reduction process. The result is a STRIPS operator that achieves the same effect as theorem proving on the CI claim: the addlist contains the result of deliberation about the augmentation action and the deletelist the resulting revision action. The STRIPS operator compiled out of **IC1** is shown in Figure 5.4.

5.5 Search Control Issues

We need abstract representations of encodings to provide good search control for instantiating CI and WI lemmas for the verification and generation of irrelevance claims. We introduce a structure called a *definedness graph* that succintly captures the definability relationships between the conceptual primitives in a particular encoding.

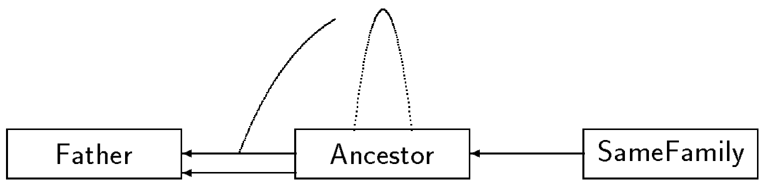


Figure 5.5: The definedness graph for encoding E_1 of the kinship problem

5.5.1 Definedness Graphs

Definition 25 *A definedness graph (V,E) of an encoding $E_{\mathcal{C}}$ of a conceptualization $\mathcal{C}$ $=\{\mathcal{O},\mathcal{F},\mathcal{R}\}$ is a directed AND-OR graph with vertex set V=$\{O,F,R\}$ where O,F and R are elements denoting $\mathcal{O},\mathcal{F},\mathcal{R}$ respectively, and the edge set $E = \{(v_i,\ \{v_{j_1},\ldots,v_{j_k}\}) \mid v_i \in V$ is defined in terms of the elements of $\{v_{j_1},\ldots,v_{j_k}\}$ in $V\}$.*

The definedness graph for the encoding E_1 of the kinship problem is shown in Figure 5.5. SameFamily is defined in terms of Ancestor. Father is a primitive relation symbol since it is a sink node in the graph. The definedness graph makes it clear that the only *role* of Father in this encoding is to define Ancestor. Ancestor itself is grounded in Father. The recursive part of the definition of Ancestor is captured by the (Ancestor,({Ancestor,Father}) edge. Notice the AND-arc that captures this graphically in Figure 5.5.

Notice that the definability relationship is established in a particular encoding. A definedness graph hides details of the actual definition, it only preserves the fact that a particular element in an encoding *can* be defined in terms of others. If for instance, the Ancestor relationship had been defined as

$$\mathsf{Ancestor}(x,y) \Leftarrow \mathsf{Father}(x,z) \land \mathsf{Ancestor}(z,y)$$

we would still have the same definedness graph as above. Thus these graphs are really abstractions of particular encodings.

The graph-theoretic interpretation of definability in terms of definedness graphs allows us to construct them given the encodings. The definability checks needed to determine if

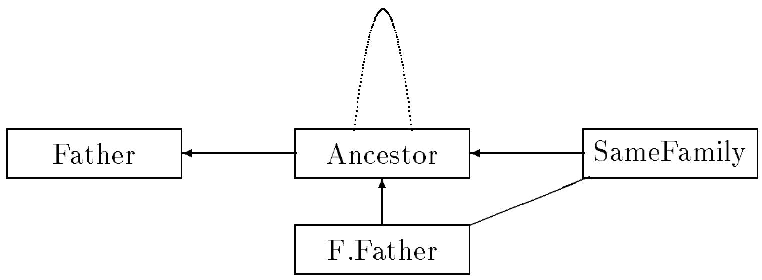


Figure 5.6: The definability map

one conceptualization makes fewer distinctions than another can be translated to path-finding problems on these graphs. We provide algorithms and complexity results for these operations.

Theorem 17 *The definability graph for Horn clause encodings can be constructed in time linear in the length of the encoding.*

Proof: We give a constructive proof. For every sentence in the encoding of the form $h \Leftarrow b_1, b_2, \ldots, b_n$, we add to E the edge (h, B) where $B = \cup_{i=1}^{n}$ if it doesn't exist there. We add h and the b_i's to the vertex set V. This construction looks at every sentence only once and is thus linear in the size of the encoding. $\Box$.

Suppose we have the encodings of two conceptualizations $\mathcal{C}_1$ and $\mathcal{C}_2$ as well as their articulation theory $\mathcal{A}$ represented as definedness graphs. Figure 5.6 shows an example. To determine whether an element c in $\mathcal{C}_1$ is definable in terms of the elements in $\mathcal{C}_2$ we check whether there is a path from elements in $\mathcal{C}_2$ to c in the definability map.

To present the path-finding algorithm, we need to distinguish three types of nodes in the definedness graph

1. Selfloop-Nodes: $v_i \in V$ that have an edge (v_i, B) where $v_i \in B$.

2. And-Nodes: $v_i \in V$ that have an edge (v_i, B) where B is a non-empty, non-singleton set.

3. Singleton-Nodes: $v_i \in V$ that have an edge (v_i, B) where B is a singleton set.

These are not mutually exclusive categories.

Theorem 18 *When the only cycles in a definedness graph G are those due to selfloop-nodes, the complexity of determining whether a node* c *in G is defined in terms of another set of nodes N in G is polynomial in the number of edges.*

Proof: We provide a constructive proof. We start with the set `In-Set` that initially contains the elements in N. We add a node v_i to this set if it is not already there when there is an edge (v_i, B) in E where $B \subseteq$`In-Set`. For an And-node this means that *all* the nodes that define it have to present in `In-Set`. We can add a selfloop-node v_i only if there is an edge (v_i, B) in E where $v_i \neg \in B$. This requirement guarantees that all recursive definitions are grounded. The addition of nodes to `In-Set` stops when c is added to `In-Set` in which case we declare that c is defined in terms of N in G, or, no more additions can be made, in which case c is not defined in terms of N in G. When G has no selfloop-nodes, this algorithm looks at every edge exactly once and is thus linear in the number of edges. If there are m selfloop-nodes then the algorithm makes m passes over the edge list in trying to satisfy the groundedness condition. When there are non-trivial cycles in the definedness graph (these correspond to mutually recursive definitions), this algorithm becomes exponential in the number of edges in G. $\square$.

To see how this algorithm works, consider the problem of determining whether FoundingFather can be defined in terms of Father in Figure 5.6. We start with `In-Set` initialized to {Father}. Since we have the edge (Ancestor, {Father}) in E, and Father is present in `In-Set`, we add Ancestor to `In-Set`. Since (Ancestor,{Ancestor}) is in E, and the groundedness condition for Ancestor is satisfied by the previous edge, we can add Ancestor to `In-Set` but it is already there! Using the edge (FoundingFather,{Ancestor}) in E, we can add FoundingFather to `In-Set` since Ancestor is already in it. Now the algorithm terminates successfully reporting that FoundingFather has a definition in terms of Father in the encoding represented by the definedness graph.

The definability map allows us to analyze the conceptualization intensionally. One way to think about the elements of a conceptualization is as the elements of a basis set. Using the construction in the theorem above, we can determine if there is redundancy in the conceptualization itself. If we can define parts of a conceptualization from other parts of it, we will call that conceptualization redundant. Efficiency issues can be discussed by cutting the conceptualization by a *p*-*d* cut. The items on the *p* side of the cut will be extensionally stored in the encoding. The items of the *d* side will be derived from the extensionally stored items. Note that the definability analysis is meta to the conceptualization itself. We can

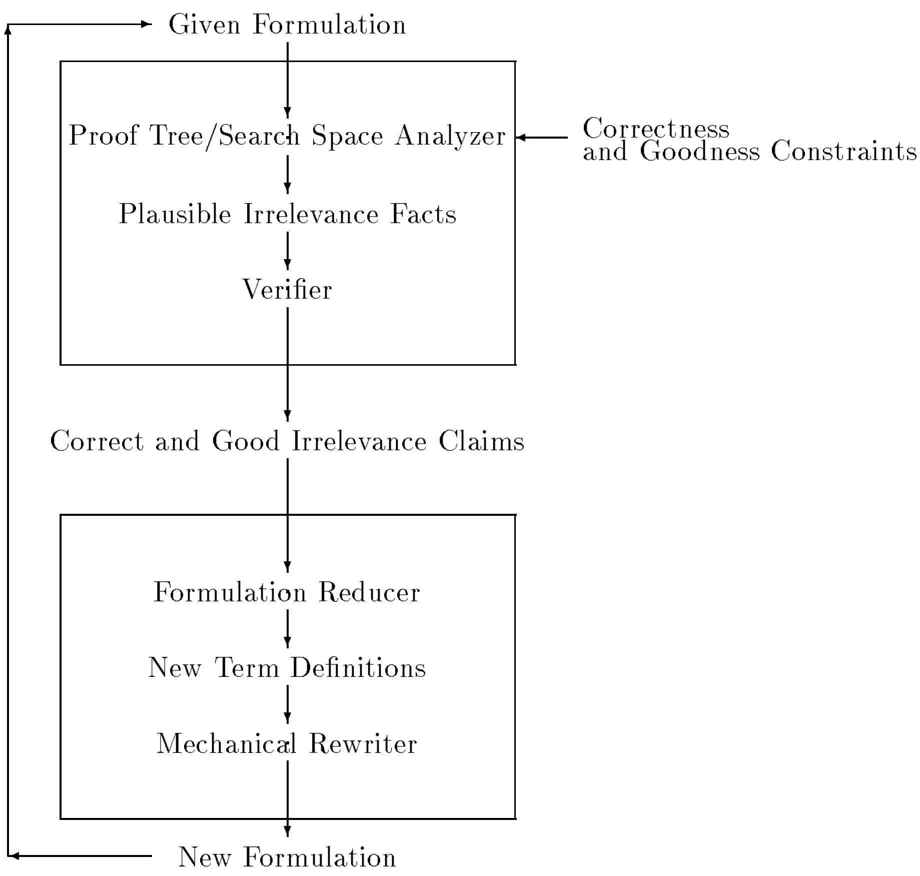


Figure 5.7: The Architecture of a Reformulator

determine a range of encoding shifts by simply defining the possible cuts of the definability map. Setting this up as a combinatorial optimisation problem is an interesting problem for future research.

5.6 The Architecture of a Reformulator

The discovery and reduction components are put together as indicated in Figure 5.7. Only the empirical method of discovery has been programmed: currently the discovery module takes the given formulation and the correctness and goodness constraints as input and analyzes the computation to generate irrelevance claims that are correct (so the goal will be preserved in the reformulation) and good (so the computation of the goal respects the computational goodness constraints). The reduction process takes the meta-theoretical irrelevance claims and either performs extensional reduction to generate the new formulation directly, or intensional reduction to produce the necessary definitions of new predicates and objects. A rewrite system then generates the new formulation in terms of these definitions. At present, the reduction component has been implemented in the meta-level programming system MRS[Gen83b]. It has been tested on the Founding Father example and a few variants of it. The Verifier has succesfully verified the irrelevance claims presented in this thesis. Extending the capabilities of the verifier requires adding powerful lemmas to the irrelevance calculus. The discovery component is implemented as a set of demons that look for regularities in the symbolic computation trace. Extending the discovery component requires programming in more general-purpose regularity detectors in computational traces. The two proof reconfiguration methods have been used to generate Thevenin equivalents of analog circuits.

Chapter 6

Conclusions and Future Work

6.1 Summary of Contributions

Research in reformulation is purely exploratory in nature because our knowledge is too poor to even formulate the appropriate questions, let alone solve them. The main contribution of this thesis is a conceptual framework in which relevant questions about reformulations for computational efficiency can be phrased and answered. The framework attempts to combine a way of looking at things that has powerful heuristic value with a collection of mathematical definitions and theorems that can be used for rigorous derivation of results. A Type 1 theory that captures regularities in the space of reformulations has been uncovered. The theory can analyze reformulations by justifying them, and in some cases generate them. The chief result from the analysis is the development of a 2-step meta-theoretic method for the automation of abstraction reformulations for computational efficiency.

1. Generate irrelevance claims that are true of the given formulation with respect to the given class of goals.

2. Reduce the formulation by the irrelevance principle.

Two general methods for the generation of irrelevance claims have been investigated: deriving them in the meta-theory by backward-chaining on the lemmas in the calculus of irrelevance, as well as by empirical analysis of proof and search spaces. Reduction methods have been developed that generate a large class of reformulations: the generation of Thevenin equivalents, the transformation of the Fibonacci computation to tail recursion, several compiler optimizations, encoding variations of the FoundingFather example, macro-operators and macro-objects.

Effective *mechanization* of this method has been accomplished on the class of abstraction reformulations called *elimination of intermediates*. The irrelevance claims for this class have a very special form: a gross step in the computation renders an intermediate step irrelevant. The meta-theoretic irrelevance lemmas of the irrelevance calculus allow us to generate these claims in a goal-directed fashion. The reductions also have a specific form: they either involve wholesale removal of relations, or the introduction of new relations and objects that are definable in terms of the existing ones.

The theory has been empirically verified by constructing a prototype of a first-principles reformulator whose architecture is shown in Figure 5.7. It has been tried on the kinship example and its variants, as well as the generation of Thevenin equivalents of analog circuits. An interesting empirical fact was that the irrelevance claims used to discover the FoundingFather relation have the same form as the ones used to derive the concept of a Thevenin equivalent of a circuit from Kirchoff's and Ohm's laws. Historically, the concept of Thevenin equivalents was discovered almost 100 years after Kirchoff's laws became known. Our reformulator derived the notion of Thevenin equivalents using the rather limited set of discovery and reduction methods at its disposal.

This shows that the framework explored in this thesis has the potential to unify disparate abstraction phenomena as instances of a powerful invariant in granularity shifts: the most economical description is the one that uses concepts of the largest granularity consistent with the correctness constraints. The ability to generate reformulations from such basic considerations can have significant impact not only on AI, but also in improving our current stock of scientific concepts and formulations.

6.2 Evaluation

6.2.1 The nature of irrelevance reformulations

Reformulations involve modifying both conceptualizations (or models) and encodings. The actual end product of a reformulation is an encoding (or theory) with better computational properties. The semantics of the theory revision is explained in terms of the effects on its models. However, not all encoding shifts can be explained in terms of models. If for instance, we replace the expression $a \wedge b$ by the expressions $b \wedge a$, this replacement might have computational impact, (a might be false most of the time and might be cheaper to compute), however it does not change the models of the theory. Conjunct orderings of this form are an instance of theory revision that is not visible at the level of models. This is not surprising, because models only characterize truth without considering the cost involved in computing truth values. In sum, reformulations at the encoding level are a description of the phenomenon at a fairly fine-grained level. The definition of reformulation at the level of conceptualizations is a coarser grained description. The irrelevance principle states that minimizing distinctions subject to the correctness constraints leads to the construction of more efficient theories.

Unfortunately, not all efficiency improvements on theories require minimization of distinctions. In fact, making algorithms efficient often involves making more distinctions. For instance, quick sort does clever bookkeeping compared to the simple minded bubble sort algorithm. The worst case time for both algorithms are the same, and a very fine grained average case complexity analysis is needed in order to explain why quicksort does better than bubble sort. This reformulation from bubble sort to quicksort is an encoding level shift as opposed to a conceptual shift, because both are sorting algorithms at the specification level. However, we can write irrelevance claims on the computation trace, the reduction of the formulation by these claims then requires making finer distinctions rather than removing them.

There are limitations introduced by the specific irrelevance discovery and reduction methods discussed in this thesis. The subset irrelevance definition is not powerful enough to explain and generate Type 2 abstractions. The set abstraction in the missionaries and cannibals is an example of this class. Refinement reformulations that involve introducing primitives that are not definable in terms of what is known, are impossible to automate using irrelevance reformulations.

6.2.2 The power of the meta-theoretic approach

The method of irrelevance minimization is an analytical technique for learning new vocabulary terms for the purpose of improving the efficiency of computation. This is done by an analysis of the task requirements and modifying old representations to meet more specific goals. The knowledge brought to bear on this process includes that of representations, the problem solver and the purpose of the representation. The calculus of WI and CI allow us intensional specification of the properties of the conceptualization and the computation. The meta-theoretic irrelevance principle then creates minimal conceptualizations with encodings that perform minimal computation. The tools for describing conceptualizations, proof and search spaces along with the well defined methods of modifying them in goal-sensitive ways are critical for making this method of reformulation feasible.

6.2.3 Issues in Validation

In his insightful commentary on the field, Marr declared that the goal of AI is to study useful information processing problems and to propose an abstract account of how to solve them. A result in AI consists of the isolation of a particular information processing problem and the statement of a method for solving it.

The problem of reformulating representations to make them computationally efficient with respect to a set of goals has been isolated. A clean Type 1 theory that describes a normative principle that governs the class of abstraction reformulations has been discovered. A partial inversion of this principle that generates abstractions automatically has also been accomplished.

The methodology adopted involved doing theory before practice because the phenomenon was too ill understood to benefit from programming. An attempt to program would only have led to the development of special methods that work in a specific domain. The goal of the thesis was to obtain an understanding of the reformulation phenomenon well enough to make it feasible to suggest good reformulations in a variety of domains. As is necessary in a ground breaking study, a fruit fly was needed to fuel the research: an example that was simple enough to do paper and pencil analyses on, and complex enough to contain the essential difficulties of automating reformulation. The kinship example was chosen because it satisfied both criteria. Since the introduction of the *FoundingFather* relation symbol is a general graph-theoretic transformation, its derivation would apply to almost any graph search problem, and all formulations of AI problems in the state space model fit this framework. So generality was not sacrificed.

How is this theory to be validated? To do this, we use the tri-step framework described in Professor Buchanan's essay on validating AI theories and systems [Buc87]. The theoretical and the analytical steps of formulating the problem, and proving that the proposed solution will work have been carried out. The empirical validation was carried out by implementing an irrelevance reformulator described in Chapter 5. The implementation of has been tested on some small examples in graph theory and analog circuits. The power of the approach is demonstrated by the fact that the reformulator was able to automatically synthesize a new conceptual primitive that is beyond the reach of reformulators that are cliche-based. This is the first existence proof of a first-principles reformulator. The scaling up of the implementation to tackle larger scale problems will require conceptual advances in the ability to describe search spaces intensionally. We intend to apply it to describe an operational amplifier at various levels of abstraction. We also wish to test the applicability of the method to non-discrete domains: in particular, the discretization of space to make path planning problems easier to solve.

The analysis of why the method works and the cases where it works is accomplished theoretically by proving appropriate theorems that establish the limits as well as the capabilities of the method. However, ablation studies have not been performed. What happens to the method if any of the conditions specified by the theorems are violated? This requires the design of good experiments on the implementation; and much thought requires to be put into the enterprise. This work will be done as a follow-up validation project.

The quantitative results in this thesis are results on reduction in size (expressed in big-O terms) of the search space by reduction methods. Some are results on the complexity of verification of irrelevance claims and the efficiency of irrelevance reasoning. The lack of good tools for measuring the impact of removal of irrelevant information affects our ability to be able to make more precise claims at this point. Much of the work has been of a descriptive rather than prescriptive in nature: the inversion of the normative irrelevance principle has been achieved on a small class of problems.

Even though this is a formal thesis in the sense defined in [Buchanan87], the theorizing was done with actual data in mind. Examples of reformulation behaviour that we wanted to capture included all shifts to formulations of coarser granularity. There is informal psychological evidence that humans are very good at this. Unfortunately, there is almost no formal psychological data gathered on reformulation behaviour in humans that is of the kind investigated in this thesis. Part of it is because the irrelevance principle is compiled into the human reasoning system and we rarely introspect about it; we simply ignore detail that is irrelevant to the present goals. However, our robots are not endowed with this mechanism and this thesis presents an analysis of the mechanism, along with ways of actually compiling it into the behaviour of robots.

There is a deep concern for the practical tractability of the reformulation process itself. Chapter 5 deals exclusively with ways of defining good representations of conceptualizations, encodings, search and proof spaces and how to use them to compile out special cases of irrelevance minimization.

6.3 Future Research

Future research plans include extending the theoretical framework by investigating a richer class of irrelevance claims that form the basis for automating approximations and inductive reformulations. The experimental component seeks to test the theoretical methods in large scale problems in engineering design and planning, with a view to extending the capabilities of present-day computer-aided design systems.

6.3.1 Theoretical Issues

1. **Extensions to the Theory of Irrelevance**

   - Approximations: The choice of primitives to describe a problem is directly related to their value in the solution of the problem. Abstraction reformulations can be seen as shifting of the focus of attention of the problem solver from the irrelevant aspects of the problem to the essential ones. The only abstractions considered so far were pure deductive ones: the reformulation did not affect the correctness, only the efficiency of inference. A large class of useful abstractions called *approximations* trade accuracy for efficiency. A classic case is the simplification of the hybrid-pi model of a transistor to the base-emitter model under the low frequency condition. We propose to automate their construction by relaxing our constraints on the specification of irrelevance claims to allow for approximate correctness. We shall then develop lemmas of approximate irrelevance and reduction schemes that will alter formulations to minimize approximate irrelevance. Our test bed will be problems in planning in the blocks-world and simplification of transistor models. The results that we hope to obtain are general-purpose approximation methods that can be used across domains to relax models to incrementally trade-off accuracy for efficiency.

- Iterative minimization methods: The methods developed in this thesis construct 1-level abstractions (of which FoundingFather is an example). We would like to extend the methods to develop abstraction hierarchies as in Abstrips. The levels would correspond to gradual ignoring of information. The basic method would be the *ordered* reduction of a formulation by irrelevance claims as opposed to the simultaneous reduction used in Chapter 4. The development of techniques to achieve this requires construction of cost models that allow for the rapid calculation of the effect of reduction by an irrelevance claim to decide on a good ordering for reduction. Our test bed is Abstrips: we will start from the most detailed description of the action operations and facts about preconditions (how easy they are to achieve), and construct a hierarchy of action operators that minimizes planning time for given classes of planning goals.
- Extending the irrelevance claims to specify probabilistic information: many irrelevance claims in the world are of a probabilistic nature. The claims describing factors that are irrelevant in a medical diagnosis task are an example. To allow their specification, we need to examine the semantics of probabilistic irrelevance. This would be pre-cursor to the development of methods to act upon these claims.

2. **Automating Refinement and Isomorphic Reformulations**

   The reasoning needed to accomplish reformulations consists of means of evaluating the epistemic and computational consequences of perturbing the conceptualization either by the introduction or the removal of a conceptual element. Irrelevance claims were a particularly nice form of justification because they tied the exclusion of a conceptual element directly to its computational consequences. Finding further justifications of this form to automate isomorphic and refinement reformulation is a logical next step in our theoretical investigations.

   - Development of further invariants in the representation-inference tradeoff: The method of irrelevance minimization allows for the introduction of new concepts that simplify inference to achieve a certain class of goals. This is done by pruning entire subtrees in the computation. A related method that we will investigate is the substitution of subtrees by simpler computations and propagating their effects into the vocabulary for the problem. The concept of substitutability will be formalized with an eye to justifying and automating the class of refinement and isomorphic reformulations.

- Inductive reformulations: the extension of these methods to cover non-deductive inference is a project begun in [RS88]. Related work on this is done under the rubrik of the study of bias in machine learning. Non-deductive systems are very sensitive to the form of the representation chosen for the premises of each inference. With one representation the system may return the correct solution, with another it may not, even though they both contain the same information. Inductive reformulations seek to transform a given representation to one that maximizes correctness of conclusions with respect to a given problem solver. Our initial focus for the study of this phenomenon will be reformulation descriptions of to make analogy by similarity (by counting features) work.
- Symmetry reformulations: Symmetry is a general kind of redundancy. If a system possesses symmetry, then the behaviour of a subpart can be computed by knowing the behaviour of a symmetric subpart and the symmetry function. For redundancy, the symmetry function is the identity function. Using the search space metaphor, symmetry is said to exist in a search tree if a subtree can be computed from another subtree plus a translation function. This amounts to reusing old computation and thus leads to computational efficiency. Automating symmetry reformulations allows us to generalize the class of irrelevance reformulations.

3. **Improving the Efficiency of Deriving Reformulations**

- Reformulation Algorithms: An approach to containing the complexity of the first-principles reasoning is to compile some of the reduction inferences into graph-theoretic algorithms. The FoundingFather transformation can be compiled down to a standard union-find algorithm. We will investigate classes of irrelevance minimization inferences that can be subject to this kind of compilation.
- Reduction Lemmas: The complexity of intensional reasoning requires that we develop reduction lemmas akin to the irrelevance lemmas to speed up the process. Work on this as well as on better intensional descriptions of proof and search spaces is critical to making the approach a practical one.

6.3.2 Experimental Projects

No amount of theoretical investigations can replace a practical implementation of the ideas being investigated, and such an implementation is a key part of the research effort we are proposing. The prototype irrelevance reformulator built in Chapter 5 automatically derives the partition representation of an equivalence relation, as well as the concept of Thevenin equivalents from Kirchoff's and Ohm's laws. This is a capability that no other AI system possesses. This reformulator is built upon the meta-level reasoning system MRS [Gen83b, Gen83a, Rus85]. MRS is extensively used in academic and industrial environments and provides its users with a variety of knowledge representation and inference procedures – it is one of the few knowledge representation languages in AI that provide constructs for *describing* representations and inference methods. This feature is critical for the development of the meta-theoretic reasoning methods for reformulation. The prototype reformulator built in this thesis uses MRS and Lisp. The extensions that we plan to make on this prototype system include:

1. **A Reformulation Assistant**
   The inflexibility of present day design environments is largely due to their inability to reason with multiple models of a domain at different granularity levels. The theory of incremental reformulations proposed here can be used to synthesize abstractions of a detailed model in a goal-directed manner. We intend to test our theory and extend the available set of reduction lemmas by building a reformulation assistant. This is a system that accepts irrelevance claims from a domain-expert and synthesizes a formulation that doesn't make the distinctions specified by the irrelevance claims.

   - With the advances being made in the technology of manufacturing digital devices it is possible to build systems of unprecedented complexity. Representing, and reasoning about such systems requires describing them at varying levels of abstraction to contain the complexity of tasks like diagnosis and test generation. This places a large burden on the specifier of a system. Also the system is constrained by the *fixed* abstraction levels provided. The methods provided in this proposal can be used to synthesize abstraction levels that are tuned to particular task requirements. We plan to test this idea in the context of the *Helios* design environment [Sin86, Gen84]. One of the first projects will be to take the specification a full adder at the gate level and derive the functional specification that abstracts details of the structure so as to make simulating the circuit extremely efficient. The examples of structural and functional abstractions presented in Chapter 2 of [Sin86] will then be automated: this will be an excellent demonstration of the power of automated reformulation.

- The abstractions in the digital circuits domain involve moving from discrete to more discrete descriptions. To test the utility of the theory in abstracting continuous phenomena: we will explore automatic discretization of space to make motion planning efficient in collaboration with Bruce Donald at the Computer Science Department at Cornell University. A first step is the expression of discretization criteria [LP83, Bro83, Don87] as irrelevance claims and the development of special-purpose reduction methods that generate tilings of a given region in 3-space.

2. **Combining First-Principles and Cliched Reformulations**
Our prototype reformulator works from first principles. The meta-theoretic reasoning required is expensive: we therefore wish to explore an architecture that allows integration of the use of previously derived reformulations (cliched reformulations) with the ability to synthesize new ones on demand. This requires generalizing a newly derived reformulation so as to increase its range of application. For instance, the FoundingFather reformulation can be generalized to be a useful reformulation for the computation of *any* equivalence relation defined in terms of a partially ordered relation. Generalization of this kind can be accomplished by using the well-established method of explanation-based generalization [MTMS86]. The novel aspect of this use of EBG is that the explanations formed and generalized are meta-theoretical.

Bibliography

[AB75] A. Anderson and J. Belnap. *Entailment: the logic of relevance and necessity.* Princeton University Press, 1975.

[AC87] Philip E. Agre and David Chapman. Pengi: An implementation of a theory of activity. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, Seattle, WA, August 1987. American Association for Artificial Intelligence.

[Ama68] S. Amarel. On representation of problems of reasoning about actions. In Donald Michie, editor, *Machine Intelligence 3*, pages 131–171. Edinburgh University Press, Edinburgh, Scotland, 1968.

[BR87] C. Beeri and R. Ramakrishnan. On the power of magic. Technical Report to appear in Journal of Logic Programming, Computer Science Department, The Hebrew University, 1987.

[Bro83] R.A. Brooks. Solving the find-path problem by good representation of free space. *IEEE Transactions on Systems, Man and Cybernetics*, 13, 1983.

[Bro87] R.A. Brooks. Intelligence without representation. In D. Kirsh, editor, *Proceedings of the Workshop on Foundations of AI*, Dedham, MA, 1987.

[Buc87] B.G. Buchanan. Artificial intelligence as an experimental science. *Synthese*, 1987.

[Car87] S. Carey. *Conceptual Change in Childhood.* MIT Press, 1987.

[CME82] Glaser R. Chi M. and Rees E. Expertise in problem solving. In R. Sternberg, editor, *Advances in the Psychology of Human Intelligence, Vol 1.* Lawrence Erlbaum, Hillsdale, N.J, 1982.

[Don87] Bruce R. Donald. *Error Detection an Recovery for Robot Motion Planning with Uncertainty.* PhD thesis, MIT, 1987.

[DR87] T. Davies and S.J. Russell. A logical approach to reasoning by analogy. In *Proceedings of IJCAI-87*, Milan, Italy, 1987. Morgan kaufmann.

[End66] Herbert Enderton. *Mathematical Logic.* Academic Publishers, 1966.

[FB85] L-M. Fu and Bruce G. Buchanan. Learning intermediate knowledge in constructing a hierarchical knowledge base. In *Proc. of IJCAI-85*, pages 659–666, Los Angeles, CA, August 1985. IJCAI, Inc.

[Fla88] N. Flann. Improving problem solving performance by example-guided reformulation of knowledge. In *Proceedings of the Workshop on Change of Representation and Inductive Bias*, New York, June 1988. Philips Laboratories.

[FN71] Richard F. Fikes and Nils J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[Gen83a] M.R. Genesereth. *A Meta-level Respresentation System.* Heuristic Programming Project, Stanford University, Stanford Ca. 94305, 1983.

[Gen83b] M.R. Genesereth. An overview of meta-level architecture. In *Proc. AAAI*, Washington, D.C., August 1983. AAAI.

[Gen84] M.R. Genesereth. The use of design descriptions in automated diagnosis. *Artificial Intelligence*, 24(1-3), December 1984.

[GN87] M.R. Genesereth and N.J. Nilsson. *Logical Foundations of Artificial Intelligence.* Morgan Kaufmann, 1987.

[Gre85] Russell Greiner. *Learning by Understanding Analogies.* PhD thesis, Stanford University, 1985.

[Har86] G. Harman. *Change in View.* MIT Press, Cambridge, MA, 1986.

[Hay81] P.J. Hayes. The logic of frames. In B.L. Webber and Nilsson N.J., editors, *Readings in Artificial Intelligence.* Morgan Kaufman, 1981.

[Hem65] Carl Gustav Hempel. *Aspects of Scientific Explanation and other Essays in the Philosophy of Science.* The Free Press, New York, 1965.

[Hob85] Jerry Hobbs. Granularity. In *Proc. of IJCAI-85*, Los Angeles, CA, August 1985. IJCAI, Inc.

[Kel87] R.M. Keller. *Learning in Context.* PhD thesis, Rutgers University, November 1987.

[Kor80] R. E. Korf. Toward a model of representation changes. *Artificial Intelligence*, 14(1):41–78, August 1980.

[Kor83] R. Korf. *Learning to solve problems by searching for macro-operators.* PhD thesis, Carnegie-Mellon University, 1983.

[Kuh87] T. Kuhn. The presence of past science. Technical Report unpublished memo, 1987.

[Len82] D.B. Lenat. The nature of heuristics. *Artificial Intelligence*, 19:189–249, 1982.

[LHRK79] D. Lenat, F. Hayes-Roth, and P. Klahr. Cognitive economy. Knowledge Systems Laboratory Report KSL-79-15, Stanford University, June 1979.

[LJP87] Newell A. Laird J.E. and Rosenbloom P.S. Soar: An architecture for general intelligence. *Artificial Intelligence*, 33:1–64, 1987.

[Low88] M. Lowry. Problem reformulation and adt. In *Proceedings of the Workshop on Change of Representation and Inductive Bias*, New York, June 1988. Philips Laboratories.

[LP83] Tomas Lozano-Perez. Spatial planning: A configuration space approach. *IEEE Transactions on Computers*, C-32, 1983.

[Man74] Z. Manna. *The Mathematical Theory of Computation.* McGraw Hill, 1974.

[Mar76a] W. Mark. *A Reformulation Model of Expertise.* PhD thesis, Massachusetts Institute of Technology, 1976.

[Mar76b] D. Marr. Artificial intelligence: a personal view. Technical Report AIM 355, AI laboratory, MIT, 1976.

[McC64] John McCarthy. A tough nut for proof procedures. Technical Report Sail AI Memo 16, Computer Science Department, Stanford University, 1964.

[McC79] J. McCarthy. First order theories of individual concepts and propositions. In *Machine Intelligence 9*. Ellis Horwood Ltd., Chichester, England, 1979. Also STAN-CS-79-724.

[MG84] J. Mackinlay and Michael R. Genesereth. Expressiveness of languages. In *Proceedings of the Fourth National Conference on Artificial Intelligence*, pages 226–232, Austin,TX, August 1984. American Association for Artificial Intelligence.

[Min86] Marvin Minsky. *Society of Mind*. Simon and Schuster, 1986.

[Min88] Steve Minton. *Learning effective search control knowledge*. PhD thesis, Carnegie-Mellon University, 1988.

[Mos81] J. Mostow. *Mechanical transformation of task heuristics into operational procedures*. PhD thesis, Carnegie-Mellon University, 1981.

[MTMB82] Utgoff P. E. Mitchell Tom M. and Banerji R. B. Learning by experimentation: Acquiring and refining problem- solving heuristics. In *Machine Learning*. Tioga, 1982.

[MTMS86] Keller R.M. Mitchell Tom M. and Kedar-Cabelli S.T. Explanation-based generalization: A unifying view. *Machine Learning*, 1(1), January 1986.

[Mug88] S. Muggleton. Constructive induction in first-order logic. In *Proceedings of the Workshop on Change of Representation and Inductive Bias*, New York, June 1988. Philips Laboratories.

[New65] A. Newell. Limitation of the current stock of ideas about problem solving. In Kent and Taulbee, editors, *Conference on Electronic Information Handling*. Spartan Books, Washington, D.C., 1965.

[NS76] A. Newell and H. A. Simon. Computer science as empirical inquiry: Symbols and search, the 1976 ¡acm¿ turing lecture. *Communications of ACM*, 19(3):113 – 126, 1976.

[Pyl84] Z. Pylyshyn. *Cognition and Computation.* Cambridge, MIT Press, 1984.

[Qui63] W.V.O. Quine. *From a Logical Point of View.* Harper and Row, New York, 1963. 2nd edition (revised).

[REFJ81] P. E. Hart R. E. Fikes and N. J.Nilsson. Learning and executing generalized robot plans. In B.L. Webber and N.J. Nilsson, editors, *Readings in Artificial Intelligence.* Morgan Kaufmann, 1981.

[Rid88] P. Riddle. An approach to learning problem decomposition schemas and iterative macro-operators. In *Proceedings of the Workshop on Change of Representation and Inductive Bias*, New York, June 1988. Philips Laboratories.

[RS88] S.J. Russell and D. Subramanian. Mutual constraints on representation and inference. In P. Brazdil, editor, *Proceedings of the Workshop on Machine Learning, Meta Reasoning and Logics*, Sesimbra, Portugal, February 1988.

[Rus85] S.J. Russell. The compleat guide to mrs. Technical Report STAN-CS-85-1080, Computer Science Department, Stanford University, 1985.

[Sac74] Earl D. Sacerdoti. Planning in a hierarchy of abstraction spaces. *Artificial Intelligence*, 5(2):115–135, 1974.

[SG87] D. Subramanian and M.R. Genesereth. The relevance of irrelevance. In *Proceedings of IJCAI-87*, Milan, Italy, August 1987.

[Shm86] O. Shmeuli. Decidability and expressiveness aspects of logic queries. Technical Report unpublished manuscript, Computer Science Department, Technion, 1986.

[Sim82] H.A. Simon. *Models of Bounded Rationality, Volume 2: Behavioral Economics and Business Organization.* Cambridge, MIT Press, 1982.

[Sin86] Narinder Singh. *Exploiting Design Morphology to Manage Complexity.* Kluwer Academic Press, Norwell, MA, 1986.

[SP88] J. Schlimmer and A. Proca. Learning, problem-solving and representation change. In *Proceedings of the Workshop on Change of Representation and Inductive Bias*, New York, June 1988. Philips Laboratories.

[Sus77] G.J. Sussman. Slices: at the boundary between analysis and synthesis. Technical Report AI Memo 433, MIT Artificial Intelligence Laboratory, 1977.

[SW89] D. Subramanian and J. Woodfill. Making situation calculus indexical. In *Proceedings of the First International Conference on Principles of Knowledge Representation*, Toronto, May 1989.

[TF85] A. Takeuchi and K. Furukawa. Partial evaluation of prolog programs and their application to meta-programming. Technical report, ICOT, 1985.

[Ull82] J.D. Ullman. *Principles of Database Systems*. Computer Science Press, 1982.

[Utg86] P.E. Utgoff. *Shift of Bias for Inductive Concept Learning*. PhD thesis, Rutgers University, 1986.

[Wel86] Daniel S. Weld. The use of aggregation in causal simulation. *Artificial Intelligence*, 30:1–34, 1986.