# Reasoning by Term Rewriting

by Michael Bulmer, B.Sc. (Hons)

submitted in fulfilment of the requirements
for the degree of Doctor of Philosophy

December 1995

Department of Mathematics
University of Tasmania

I declare that this thesis contains no material which has been accepted for the award of any other degree or diploma by the University or any other institution, and that, to the best of my knowledge and belief, this thesis contains no material previously published or written by another person, except when due reference is made in the text of the thesis.

Michael Bulmer

# Abstract

We propose a broad system for reasoning by term rewriting. Our general aim is to capture mathematical and scientific reasoning in a coherent system. To this end we introduce several new processes which allow concrete descriptions of standard notions.

For deductive reasoning we extend traditional methods for finding canonical rewrite systems to a general method for systems involving both equations and inequations. We introduce the notion of side conditions for non-theorems and show how they provide a new kind of meta-reasoning whereby an automated reasoner can determine why it failed to prove a given statement. A method for the automatic proof of inductive theorems by an analogue of mathematical induction is also presented.

A new algorithm is given for inductively generating conjectures (function equations) from a set of observations (a rewrite database). This is a process of scientific induction and we prove some fundamental results linking it to mathematical induction. Comparisons are given with standard inductive learning systems, such as FOIL, to illustrate the expressive power of our algorithm.

We obtain probabilistic measures of the strength of a single conjecture using statistical testing and an information measure. For a collection of conjectures we are then able to quantify Popper's well-known falsifiability criterion for the strength of a theory. We also introduce a non-standard modal operator to extended our deductive reasoning to reasoning with conjectures.

We use belief dynamics as the framework of an implementation of the reasoning methods. Consistency analysis, using the same canonical-form algorithm introduced earlier, allows the reasoner to build a belief set from given knowledge and to form a working theory from the conjectures it makes. Again a meta-reasoning is introduced, with the reasoner then able to decide what experiments need to be carried out when it conjectures more than one consistent theory from given set of observations.

Dialogues with the reasoner, generated by a prototype implementation of the work in the thesis, are given to illustrate its behaviour and the links between the internal language it uses and natural language.

# Contents

# List of Examples

# List of Figures

# List of Tables

# Acknowledgments

I would like to deeply thank Desmond Fearnley-Sander for his guidance and patient supervision during this work. His great idealism and sense of purpose have been an inspiration throughout.

Simon Wotherspoon, Tim Stokes, and La Monte Yarroll have been invaluable both in discussions of the material and in their friendship. I would also like to thank Kym Hill for his technical assistance, and all of the members of the Mathematics Department for its rich atmosphere. Finally, I give special thanks to my wife, Bronwyn, for her constant support.

CHAPTER 1

# Introduction

Both automated reasoning and term rewriting have a rich history. However, much of the focus in automated reasoning has been on methods based on classical logic and logic programming systems. Although some applications of term rewriting have been considered, such as in [**26**], it still remains a small part of the field. In this work we seek to present a general and unified approach to automated reasoning through term rewriting.

Our broad aims are then two-fold. Firstly we give a thorough set of algorithms for reasoning by term rewriting. These include completion algorithms for systems of equations and inequations, and an algorithm for generating function equations inductively from a ground rewrite system. The second aim is to use this concrete system of rewriting methods as a means of defining and exploring standard notions of deductive and inductive reasoning. Of course these aims are intimately dependent on each other. The result will be a working prototype which is both motivated by and useful for reasoning. With this in mind we will often talk about a *reasoner*, meaning simultaneously the idealized model of reasoning and the resulting concrete implementation.

We begin in Chapter 2 by giving the language in which we represent our knowledge and with which we reason. Term equations are a natural means of meeting these two needs. In order to concentrate on reasoning as much as term rewriting we adopt the simplest algebra of terms in which everything is ground (that is, variable-free), as given by Fearnley-Sander [**15**]. This language is still very expressive and also has a number of theoretical advantages, such as the decidability of termination for a given rewrite system [**12**].

An important addition to the basic language of terms is the set constructor, a special instance of the *potential entities* described in [**4**]. The role of function terms can be quite limiting as confluence requires each function to have no more than one value. Sets provide a convenient method of handling multi-valued functions.

Chapter 3 looks at automated deductive proof techniques. The traditional method of deciding whether an equation is an equational consequence of a system $E$ is to find a canonical rewrite system, $R$, for $E$ and reduce both

sides of the equation with $R$. We call such a method *proof by normalization* and note that it is limited in a number of ways. It gives no procedure for generalizing to proving inequations from systems of equations and inequations, and will also fail to prove a valid equation if the initial stage of finding the canonical system does not terminate.

We instead concentrate on *proof by contradiction* whereby we add the negation of the equation to be proved to the system and seek a canonical form of the extended system. If in doing so we find that the system is inconsistent then we say that the equation is proved. This procedure is then immediately *semidecidable* since every valid equation will eventually give such a contradiction if the completion algorithm maintains its system in reduced form at each stage. We can also immediately prove inequations, their negations now being equations that get added to the system of hypotheses.

If the completion of the extended system terminates but does not find an inconsistency then we can interpret the result as *side conditions*, conditions which can be added to the original hypotheses to produce a theorem which can be proved. This gives the automated deductive reasoner a level of meta-reasoning, whereby it can not only prove a valid theorem but can also identify why it could not prove a non-theorem.

Related to the identification of side conditions is the process of solving a system of equations for an unknown. While we do not have variables in our term language, we can treat certain terms as being indeterminate in meaning. This gives a process for finding values for such unknowns which does not require any unification procedure.

The algorithm used for proof by contradiction processes single equations and inequations only. However, we extend the proof method to a full propositional language in which the equations are the atoms. Reduction of the hypotheses and conclusion to disjunctive and conjunctive normal forms, respectively, generates a collection of subtheorems required to prove the original theorem. In Chapter 4 we show that this extension of the proof method to the propositional language gives a *consequence relation*, and hence forms the reasoning component of an AGM *belief system* [**20**].

Throughout we treat our language of a ground term algebra and the process of term rewriting as fundamental elements of reasoning. The variable-free language gives a trivial semantics for our proof techniques, placing the emphasis instead on the syntactic processing carried out by rewriting. Standard notions of classical logic, especially that of consistency, are then introduced in terms of the rewriting algorithms. Similarly, rather than looking at questions of logical completeness, we instead progressively extend the basic *canonical form* algorithm of Knuth and Bendix [**32**] to mirror such questions.

In particular, we give an algorithm which is proven to give a unique reduced form for equivalent systems of equations and inequations. This gives rise to a third deductive proof technique, dubbed *proof by invariance*, whereby a conclusion is a consequence of a system if the canonical forms of the system and the system augmented by the (non-negated) conclusion are the same.

The automated proofs so far have all been of deductive theorems, but we can also automate the proof of inductive theorems, where the conclusion is an inductive consequence of the hypotheses. This is the induction of *mathematical induction*, for which *proof by consistency* is a well known approach. We formalize it in our theory of term algebras and are able to characterize inductive theorems and the unambiguity property of Kapur and Musser [**31**] needed for proof by consistency to be valid.

A second kind of induction is that of *scientific induction* where we make observations and then conjecture new theorems based on the observations. We present an algorithm in Chapter 5 which generates function equations from a given rewrite system, the rewrite system capturing the observations made together with other general knowledge. We are then able give results which make concrete the relation between the mathematical induction and this scientific process.

Because of our simple term language we are able to quantify notions of belief using probability theory. We initial look at several probability measures of the *strength* of a single conjecture, based on statistical hypothesis testing and on the amount of information present in the conjecture. Later when looking at a system of conjectures, a *theory*, we extend these notions to quantify Popper's *falsifiability* measure of the strength of a theory [**38**].

We also look at qualitative information about belief, particularly the consistency of a belief system. In Chapter 4 we apply these ideas to systems based on declarative knowledge, where the reasoner is told information which it must then decide whether or not to believe in. We can use a close-minded model where any information that is inconsistent with already established beliefs is rejected, giving a monotonicity of belief. Alternatively, we can treat all declared knowledge as mutable, so if inconsistent information is given then we may instead reject earlier held beliefs.

When we move to inductive reasoning we must then make similar decisions about our inductively generated knowledge. In this case all of our knowledge is contingent on the observations from which it was conjectured and so if the total set of conjectures is inconsistent then we must find some subset to take as a *working theory*. Another kind of meta-reasoning arises here with the reasoner also being able to decide what is causing the inconsistency between possible theories. The result is that the reasoner can decide what *experiments* should be carried out to resolve the inconsistency and establish a single conjectured theory.

These reasoning processes then involve two types of belief dynamics, the declarative dynamics of information given to the reasoner by an external source and the inductive dynamics of the reasoner's own internal conjectures. These two processes are illustrated in Chapters 4 and 5, respectively, through the use of dialogues. These dialogues are the results from a prototype implementation of the reasoner described in this work and show the various types of reasoning and meta-reasoning in action.

CHAPTER 2

# Preliminaries

## 2.1. Language

The language we will use is that of variable-free terms. The first section here defines this algebra in terms of a generating signature. In subsequent sections we introduce structural relations between certain terms in the language, capturing these relations as rewrite rules.

### 2.1.1. Algebraic Definition

A *signature* $\Sigma$ is a pair $\langle S, \Sigma_{S^* \times S^*} \rangle$, where $S$ is a non-empty set of *sorts* and $\Sigma_{S^* \times S^*}$ is a family of sets, not necessarily disjoint, indexed by $S^* \times S^*$, where $S^*$ is the set defined by

1. $\mathsf{ground}, \mathsf{sentence} \in S^*$;
2. If $s \in S$ then $s \in S^*$;
3. If $s, t \in S^*$ then $s \times t \in S^*$;
4. If $s \in S^*, s \neq \mathsf{ground}$, then $\{s\} \in S^*$.

An element $f$ of $\Sigma_{\langle \sigma, \tau \rangle}$ is an *operator* of *type* $\langle \sigma, \tau \rangle$, and we say $f$ has *domain* $\sigma$ and *codomain* $\tau$. If $f$ has type $\langle \sigma, \tau \rangle$, we will write $f : \sigma \longrightarrow \tau$. If an operator $f$ has type $\langle \mathsf{ground}, \tau \rangle$ we will also write $f \in \tau$ and say that $f$ is a *grounded* term of type $\tau$.

We additionally require that each set $\Sigma_{\langle \sigma, \sigma \rangle}$ contains the *identity* operator $i_\sigma : \sigma \longrightarrow \sigma$, that each set $\Sigma_{\langle \sigma, \tau \rangle}$ contains the *empty set* $\phi : \sigma \longrightarrow \tau$, and that each $\Sigma_{\langle \sigma, \mathsf{ground} \rangle}$ is a one-element set containing the *erase* operator $!_\sigma : \sigma \longrightarrow \mathsf{ground}$.

We also identify the sort $\{\{\sigma\}\}$ with the sort $\{\sigma\}$, as expressed in the structural rule S7 given in Section 2.3.3.

A signature $\Sigma$ generates the *term algebra* $T_\Sigma$. The terms of $T_\Sigma$ are defined and constructed by:

1. Each operator $f \in \Sigma_{\langle \sigma, \tau \rangle}$ is a term of type $\sigma \longrightarrow \tau$;
2. If $f$ is a term of type $\sigma \longrightarrow \tau$ then $f$ is also a term of type $\{\sigma\} \longrightarrow \{\tau\}$;

3. If $t_1, \ldots, t_n$ are terms of type $\sigma \to \sigma_1, \ldots, \sigma \to \sigma_n$ then the *tuple* $\langle t_1, \ldots, t_n \rangle$ is a term of type $\sigma \to \sigma_1 \times \cdots \times \sigma_n$;
4. If $t_1, \ldots, t_n$ are terms all of type $\sigma \to \tau$ then the *set* $\{t_1, \ldots, t_n\}$ is a term of type $\sigma \to \tau$;
5. If $t$ is a term of type $\sigma \to \tau$ and $f$ is a term of type $\tau \to \psi$ then the *application* $ft$ is a term of type $\sigma \to \psi$.

If $f : \sigma \to \tau$ is a term with $\sigma$ not **ground** then we call $f$ a $\Sigma$-*function*. If $f$ is term arising from 1. then we call $f$ a $\Sigma$-*word*.

### 2.1.2. Categorical Definition

For a given $S$-sorted signature $\Sigma$, we have an associated category $\mathcal{C}_\Sigma$ with

- objects $S^*$,
- morphisms consisting of the operators in $\Sigma_{S^* \times S}$,
- a function *dom* defined by $dom(f) = \sigma$ if the morphism $f$ arose from an operator in $\Sigma_{\langle \sigma, \tau \rangle}$,
- a function *cod* defined by $cod(f) = \tau$ if the morphism $f$ arose from an operator in $\Sigma_{\langle \sigma, \tau \rangle}$,
- an identity morphism $i_\sigma$ for each object $\sigma$, corresponding to the identity operator $i_\sigma \in \Sigma_{\langle \sigma, \sigma \rangle}$, and
- composition $\circ$ given by term application defined above.

Additionally the object **ground** is a terminal object for the category, with $!_\sigma$ the unique morphism from $\sigma$ to **ground**.

The terms of our language are now generated by composition in this category, that is, by "following the arrows". This gives a simple pictorial way of describing a signature and the language it generates. We will usually use juxtaposition in place of $\circ$.

The category will have products if the signature contains any operators of arity greater than one. Much can be achieved though with only unary operators.

## 2.2. Predicates

In addition to the terminal object **ground**, each $\Sigma$ also has the distinguished object **sentence**. An element of **sentence** is to be thought of as a truth value, while a morphism from a sort $\sigma$ to **sentence** is a *predicate* on $\sigma$. We always have the two elements $\mathsf{True}, \mathsf{False} \in$ **sentence**.

We further extend our signature $\Sigma$, and the term algebra it generates, by introducing for each sort $\sigma$ the new morphism

$$=_\sigma : \sigma \times \sigma \to \textbf{sentence}.$$

Consider the signature

$$\Sigma = \left\{ \begin{array}{l} \mathsf{Alice}, \mathsf{John} \in \mathsf{person}, \\ \mathsf{father} : \mathsf{person} \to \mathsf{person} \end{array} \right\}.$$

We can then represent the equality of the terms **father Alice** and **John** as the single term

$$=_{\mathsf{person}} \langle \mathsf{father\ Alice}, \mathsf{John} \rangle.$$

We can similarly define the new morphisms $\neq_\sigma$ and $\to_\sigma$ to represent inequations and rules. We will usually write $=_\sigma$, $\neq_\sigma$, and $\to_\sigma$ as infix operators. The $\sigma$ is also determined by the codomains of the equated terms (which naturally must be same) and so we will usually write $=_\sigma$ as the polymorphic $=$.

Note that if $f, g : \tau \to \sigma$ then $=_\sigma \langle f, g \rangle : \tau \to \mathsf{sentence}$ so that $\mathrm{dom}(f =_\sigma g)$ is $\tau$. In particular, if $\mathrm{dom}(f = g)$ is **ground**, as in **father Alice** $=$ **John**, we call $f = g$ a *grounded equation.*

For a given $\Sigma$ we define the sets of $\Sigma$-*equations*, $\mathcal{E}_\Sigma$, and $\Sigma$-*rules*, $\mathcal{R}_\Sigma$, by

$$\mathcal{E}_\Sigma = \{s =_\tau s \mid s, t \in T_\Sigma,\ s, t : \sigma \to \tau\}$$
$$\mathcal{R}_\Sigma = \{s \to_\tau t \mid s, t \in T_\Sigma,\ s, t : \sigma \to \tau\}.$$

For any $\Sigma$-equation $e$ of the form $s = t$ we define the negation $\neg e$ of $e$ to be $s \neq t$. The set of $\Sigma$-inequations, $\neg\mathcal{E}_\Sigma$, is the set of negations of all $\Sigma$-equations. We write $\mathcal{F}_\Sigma = \mathcal{E}_\Sigma \cup \neg\mathcal{E}_\Sigma$ for the set of all $\Sigma$-equations and $\Sigma$-inequations. We will call a subset $F \subseteq \mathcal{F}_\Sigma$ a $\Sigma$-*system.*

Note that we also have an empirical notion of equality, where we treat the terms as strings. In particular, for $s, t \in T_\Sigma$ we write $s \equiv t$ if $s$ and $t$ are identical (as strings), and $s \not\equiv t$ if $s$ and $t$ are not identical.

To express conjunctions we introduce

$$\mathsf{and} : \mathsf{sentence} \times \mathsf{sentence} \to \mathsf{sentence}$$

so that we may, for instance, form the term

$$\mathsf{and}\langle \mathsf{father\ Alice} =_{\mathsf{person}} \mathsf{John}, \mathsf{male\ father} =_{\mathsf{sentence}} \mathsf{True}\ ! \rangle.$$

We similarly define **or**, **xor**, **implies**, and **not**.

### 2.2.1. Entities

We follow the fundamental work of Fearnley-Sander [15] and identify two important uses for grounded $\Sigma$-words in our variable-free language. Firstly we may declare that a grounded word is to represent an *entity* in the modelled world. For example, we represent the two classical notions of truth as the grounded words **True**, **False** $\in$ **sentence**, or three different people as

Alice, John, Paul $\in$ person. We call such declared words the $\Sigma$-*entities* of $\Sigma$. The products and sets of entities are also automatically entities.

The important point is that these entities are distinct. Having declared a collection of $\Sigma$-entities we implicitly require that any $\Sigma$-system $F$ contains the inequation $a \neq b$ for all $\Sigma$-entities $a$ and $b$ with $a \not\equiv b$. Thus if we declare True and False to be $\Sigma$-entities then every $\Sigma$-system will contain True $\neq$ False. We will usually declare ground words to be entities by writing them with an uppercase letter (reflecting the similarity with a proper noun).

Any grounded word that is not declared to be a $\Sigma$-entity is called a $\Sigma$-*indeterminate*. Without the implicit entity inequations it is possible to have a consistent system with x = John, for example. Indeterminates thus serve the purpose of variables in the sense of solving equations, rather than in a unification process (see Section 3.7).

Indeterminates should always be ordered higher than entities. For example, the equation x = John oriented as x $\rightarrow$ John makes sense as a variable assignment whereas John $\rightarrow$ x does not conform to our notion of entity. It is possible to view any grounded term as an indeterminate. The rule father Alice $\rightarrow$ John in a sense assigns the value John to the indeterminate father Alice.

## 2.3. Structural Rules

The structural properties of our definitions so far can be captured using rewrite rules. Throughout our work we will assume that the rules given in the following sections are available for reducing a term to normal form.

### 2.3.1. Category without Products

The only rules we require for a category without products are those for the identity and erase morphisms. These are given in [**28**] and are as follows:

**S1:** $i\, f \rightarrow f$
**S2:** $f\, i \rightarrow f$
**S3:** $f\, !\, g \rightarrow \begin{cases} f, & \text{if } \mathrm{dom}(g) = \textsf{ground}; \\ f\, !, & \text{if } \mathrm{dom}(g) \neq \textsf{ground} \end{cases}$

### 2.3.2. Category with Products

We only add one additional rule for dealing with products:

**S4:** $\langle f, g \rangle h \rightarrow \langle fh, gh \rangle$

The inclusion of projection operators can also help model certain reasoning tasks, but their use is unnecessary for our purposes.

### 2.3.3. Sets

A *potential entity* is an algebraic construction that models a case statement and their inclusion in rewriting systems greatly increases the reasoning that can be performed. For example, the potential entity

$$\mathsf{case}((\alpha, \mathsf{f}), (\beta, \mathsf{g}), (\delta, \mathsf{h}))$$

is *potentially* f, g, or h, depending on the value of the boolean conditions $\alpha$, $\beta$, and $\delta$. These general constructions, originating in [**15**], have a rich algebraic structure [4]. However, a restricted form of them motivates a construction for *sets* of terms.

Consider the example of the function **aunt** which models the function that returns the aunt of a person. Since a person may have more than one aunt, this function is multi-valued and hence cannot be directly captured by a confluent rewrite system. Potential entities in which the boolean conditions are completely indeterminate may be viewed as sets.

For example, as a potential entity we could write the function **aunt** as

$$\mathsf{aunt} = \mathsf{case}((\alpha, \mathsf{sister\ father}), (\beta, \mathsf{sister\ mother}))$$

or, using set notation instead, as

$$\mathsf{aunt} = \{\mathsf{sister\ father}, \mathsf{sister\ mother}\}.$$

The elements of the set must all be of the same type $\sigma \to \tau$, giving a set of type $\sigma \to \{\tau\}$. The structural rules for sets are the same as those for potential entities [4], suppressing the boolean conditions:

**S5:** $f\ \{g_1, ..., g_n\} \to \{fg_1, ..., fg_n\}$
**S6:** $\{f_1, ..., f_n\}\ g \to \{f_1 g, ..., f_n g\}$
**S7:** $\{\ldots, \{g_1, ..., g_m\}, \ldots\} \to \{\ldots, g_1, ..., g_m, \ldots\}$
**S8:** $\{f, f, g_1, ..., g_n\} \to \{f, g_1, ..., g_n\}$
**S9:** $\langle\{f_1, \ldots, f_n\}, g\rangle \to \{\langle f_1, g\rangle, \ldots, \langle f_n, g\rangle\}$
**S10:** $\langle f, \{g_1, \ldots, g_m\}\rangle \to \{\langle f, g_1\rangle, \ldots, \langle f, g_m\rangle\}$

S5 and S6 define the relationship between $f : \sigma \to \tau$ and $f : \{\sigma\} \to \{\tau\}$. The next four involve alterations to the boolean conditions in the **case** construction, but these are not explicitly of interest when looking at sets. Our sets are commutative since the order of terms in potential entities is irrelevant.

One structural rule we do not adopt is the identification of $\mathsf{case}((\mathsf{True}, x))$ with $x$. This rules makes algorithmic sense but results in type conflict and some reasoning difficulties. The reduction of a term by a rewrite system

ought to be type-preserving, which is not the case if we replace $\{f\}$ by $f$. Related to this is an ambiguity in the associated equational reasoning. If

$$F = \{\mathsf{aunt\ Alice} = \{\mathsf{Jenny}\}, \mathsf{aunt\ Alice} = \{\mathsf{Kelly}\}\},$$

then we have $\{\mathsf{Jenny}\} = \mathsf{aunt\ Alice} = \{\mathsf{Kelly}\}$, indicating that $F$ is inconsistent. If we allow $\{f\} = f$ then we also have

$$\begin{aligned}\mathsf{aunt\ Alice} = \{\mathsf{aunt\ Alice}\} &= \{\mathsf{aunt\ Alice}, \mathsf{aunt\ Alice}\} \\ &= \{\{\mathsf{Jenny}\}, \{\mathsf{Kelly}\}\} = \{\mathsf{Jenny}, \mathsf{Kelly}\},\end{aligned}$$

giving a second equational meaning of $F$. Hence we exclude $\{f\} = f$ so that if Alice indeed has two aunts then we must give the single equation $\mathsf{aunt\ Alice} = \{\mathsf{Jenny}, \mathsf{Kelly}\}$. The implications of this are seen again in Example 3.33. It is possible for an implementation to invoke a preprocessor to convert $F$ into $F' = \{\mathsf{aunt\ Alice} = \{\mathsf{Jenny}, \mathsf{Kelly}\}\}$. A practical reasoner should be able to learn $\mathsf{aunt\ Alice} = \{\mathsf{Jenny}\}$ and the later learn $\mathsf{aunt\ Alice} = \{\mathsf{Kelly}\}$ without giving inconsistency, particularly under the assumption that its data is noiseless. However, this is a meta-reasoning and not part of our equational theory.

Working with the boolean conditions of potential entities requires binary boolean operators, and hence products. Another advantage of the above structural rules is that by not manipulating the conditions, we can work with sets without the necessary presence of products.

EXAMPLE 2.1 (Set reduction). As an example of the structural rules S5-S8, consider the signature

$$\Sigma = \left\{ \begin{array}{l} \mathsf{Alice}, \mathsf{John}, \mathsf{Jill}, \mathsf{Jenny}, \mathsf{Kelly}, \mathsf{Mary} \in \mathsf{person}, \\ \mathsf{father} : \mathsf{person} \to \mathsf{person}, \\ \mathsf{sister}, \mathsf{aunt} : \mathsf{person} \to \{\mathsf{person}\}, \\ \mathsf{female} : \mathsf{person} \to \mathsf{sentence} \end{array} \right\},$$

and rewrite system

$$R = \left\{ \begin{array}{l} \mathsf{aunt} \to \mathsf{sister}\ \{\mathsf{father}, \mathsf{mother}\}, \\ \mathsf{father\ Alice} \to \mathsf{John},\ \mathsf{mother\ Alice} \to \mathsf{Jill}, \\ \mathsf{sister\ John} \to \{\mathsf{Jenny}\},\ \mathsf{sister\ Jill} \to \{\mathsf{Kelly}, \mathsf{Mary}\}, \\ \mathsf{female\ Jenny} \to \mathsf{True},\ \mathsf{female\ Kelly} \to \mathsf{True}, \\ \mathsf{female\ Mary} \to \mathsf{True} \end{array} \right\}.$$

Finding the value of female aunt Alice makes use of all of the given rules not involving products:

$$\begin{aligned}
& \mathsf{female\ aunt\ Alice} \\
\to_R\ & \mathsf{female\ sister\ \{father, mother\}Alice} \\
\to_{S6}\ & \mathsf{female\ sister\ \{father\ Alice, mother\ Alice\}} \\
\to_R\ & \mathsf{female\ sister\ \{John, Jill\}} \\
\to_{S5}\ & \mathsf{female\ \{sister\ John, sister\ Jill\}} \\
\to_R\ & \mathsf{female\ \{\{Jenny\}, \{Kelly, Mary\}\}} \\
\to_{S7}\ & \mathsf{female\ \{Jenny, Kelly, Mary\}} \\
\to_{S5}\ & \mathsf{\{female\ Jenny, female\ Kelly, female\ Mary\}} \\
\to_R\ & \mathsf{\{True, True, True\}} \\
\to_{S8}\ & \mathsf{\{True\}}
\end{aligned}$$

In $\Sigma$ the type of female is person $\to$ sentence, but the construction of $T_\Sigma$ included the term female : {person} $\to$ {sentence}. This is the morphism used in the term female aunt Alice. The change can be made explicitly through type inference, or implicitly, as above, through the rules S5 and S6. $\diamond$

We can view equality as a binary operator so that we may write, for example, a term

$$= \langle \mathsf{aunt, sister\ \{father, mother\}} \rangle.$$

This then involves a product of sets, and so is structurally reduced to the set of equations

$$\{\mathsf{aunt = sister\ father, aunt = sister\ mother}\}.$$

Equations involving internal set constructors thus express (non-exclusive) disjunction, as opposed to the standard sets of equations, our equational systems, which we view as conjunctions.

However, this structurally reduced form is less amenable to rewriting than that used in the $R$ of Example 2.1. We will usually view equality and rewrite as being *external* to $T_\Sigma$, maintaining $=$ and $\to$ at the head of term expressions.

### 2.3.4. Term Orderings

Let $S$ be the set of structural rules S1-S10. A term $s$ is in *structural normal form* if $s$ cannot be reduced by $S$. A term $s$ is the structural normal form of a term $t$ if $t \to_S^* s$ and $s$ is in structural normal form.

DEFINITION 2.1 (Length). The *length* $\delta(t)$ of a term $t$ with structural normal form $s$ is the number of applications in $s$ not occurring in a product or set.

For example,

$$\delta(\mathsf{father\ Alice}) = 2,$$
$$\delta(\mathsf{and}\langle\mathsf{male\ Alice}, \mathsf{True}\rangle) = 2,$$
$$\delta(\{\mathsf{Alice}, \mathsf{Lucia}\}) = 1.$$

Since sets distribute to both the right and the left, any term involving a set will have length 1.

DEFINITION 2.2. Given an ordering $>$ on the words of $\Sigma$, we extend $>$ to $T_\Sigma$ by giving an ordering on terms $s, t \in T_\Sigma$ in structural normal form.

- If $\delta(s) \neq \delta(t)$ then $s > t$ if and only if $\delta(s) > \delta(t)$;
- if $\delta(s) = \delta(t)$ then
  - if $s = \langle s_1, \ldots, s_n\rangle, t = \langle s_1, \ldots, s_n\rangle$ and $k > 0$ is the first integer such that $s_k \neq t_k$ then $s > t$ if and only if $s_k > t_k$;
  - if $s = \{s_1, \ldots, s_n\}, t = \{s_1, \ldots, s_m\}$, where the $s_i$ and $t_j$ are ordered by $>$, then
    - if $n \neq m$ then $s > t$ if and only if $n > m$;
    - otherwise if $k > 0$ is the first integer such that $s_k \neq t_k$ then $s > t$ if and only if $s_k > t_k$;
  - otherwise if $s = s_1 \cdots s_n, t = t_1 \cdots t_n$ and $k > 0$ is the first integer such that $s_k \neq t_k$ then $s > t$ if and only if $s_k > t_k$;

Thus for example

$$\mathsf{father\ Alice} > \mathsf{John},$$
$$\mathsf{and}\langle\mathsf{male\ Alice}, \mathsf{True}\rangle > \mathsf{and}\langle\mathsf{True}, \mathsf{male\ Alice}\rangle,$$
$$\{\mathsf{father\ Alice}, \mathsf{Paul}\} > \{\mathsf{John}, \mathsf{Paul}\}.$$

Because this ordering refers to the length of terms we call it a *total degree*, or *graded lexicographic*, term ordering. It will be the predominant ordering in our work since any rewrite system based on it will preserve or reduce term length for every rewrite and so must be terminating. An alternative ordering can be obtained by dropping the length conditions and filling out terms with the identity (ordered below all other words) where necessary for comparisons. This *lexicographic* ordering can sometimes be more natural than the graded ordering, but termination then needs to be verified (either empirically, or by using a decision procedure such as that given in [**12**]).

CHAPTER 3

# Proof Methods

In this chapter we are primarily interested in developing a complete proof method for deductive theorems over the propositional language with equations as atoms and connectives $\{\neg, \vee, \wedge, \rightarrow, \equiv, \oplus\}$. We find that the most amenable method is proof by contradiction, though an associated process, proof by invariance, will also play an important role. Along the way we will also examine the proof of inductive theorems, look at how non-theorems can be extended to give theorems, and finally consider how we might solve systems of equations in a variable-free language.

## 3.1. Equational Proof

DEFINITION 3.1. Let $F$ be a $\Sigma$-system. We define the relation $=_F$ on $T_\Sigma$ as follows:

**Axiom:** If $(s = t) \in F$ then $s =_F t$.
**Reflexivity:** $s =_F s$ for all $s \in T_\Sigma$.
**Symmetry:** If $s =_F t$ then $t =_F s$.
**Transitivity:** If $s =_F t$ and $t =_F u$ then $s =_F u$.
**Application:** If $s =_F t$ then $su =_F tu$ for all $u \in T_\Sigma$ with $\mathrm{cod}(u) = \mathrm{dom}(t)$.
**Product:** If $s =_F t$ then $\langle \ldots, s, \ldots \rangle =_F \langle \ldots, t, \ldots \rangle$.
**Set:** If $s =_F t$ then $\{\ldots, s, \ldots\} =_F \{\ldots, t, \ldots\}$.
**Structure:** If $s \rightarrow t$ by the structural rules S1 - S10 then $s =_F t$.

If $s =_F t$ then we write $F \models (s = t)$ and say that $s = t$ is a *consequence* of $F$.

DEFINITION 3.2. If $s =_F t$ for some $(s \neq t) \in F$ then $F$ is an *inconsistent* $\Sigma$-system, and we write $F \models \bot$. Otherwise $F$ is said to be a *consistent* system.

A *rewrite system* $R$ over $T_\Sigma$ is a subset of the rewrite rules $\mathcal{R}_\Sigma$. The *application* of a rule $(l \rightarrow r) \in R$ on a term $s$ involves replacing an occurrence of $l$ in $s$ (if any) with $r$. If the resulting term is $t$ we say that $s$ *rewrites* to $t$ by $R$ and write $s \rightarrow_R t$. If $s \rightarrow_R s_1 \rightarrow_R \cdots \rightarrow_R t$ then we write $s \rightarrow_R^* t$.

If $s \rightarrow_R^* t$ and $t$ cannot be rewritten by $R$ then we say that $t$ is an *R-normal form* of $s$. If there is some $u$ such that $s \rightarrow_R^* u$ and $t \rightarrow_R^* u$ then we write $s \leftrightarrow_R^* t$.

If there is no infinite rewriting $s \rightarrow_R s_1 \rightarrow_R \cdots$ then we say that $R$ is *terminating*. Termination is a decidable property for our ground language $T_\Sigma$ [**12**].

## 3.2. Proving Equations

Our first goal is to outline three methods for proving that a specific equation is the logical consequence of a set of equations. These methods will then be used as the basis for the more general problem of proving an equation or inequation to be a consequence of a system of equations and inequations.

The first method, proof by *normalization*, is a standard equational proof technique. Relying on the soundness and completeness of the Knuth-Bendix procedure, we prove an equational conclusion by normalizing it with respect to a canonical rewrite system generated from the equational hypotheses. This method gives no generalization to systems involving inequations, and will fail to prove a valid equation if the canonical system is unbounded. This lack of semidecidability has been addressed in [**13**], but here we avoid it by introducing a second method, proof by *contradiction*. For such a proof we augment the hypotheses with the negated conclusion and then reduce the resulting system. This requires an extension to the Knuth-Bendix procedure to process inequations.

This extended procedure is sufficient to prove anything by contradiction that could be proved by normalization, in addition to allowing proof from hypotheses involving inequations. However, for equations alone we also have the strong result that two systems are equivalent if and only if their canonical forms are identical. This gives rise to a third technique, proof by *invariance*. Our final task is to obtain a canonical form algorithm for systems with inequations which has this same property.

### 3.2.1. Proof by Normalization

The standard rewrite method for determining whether $F \models f$ is to find a canonical set of rewrite rules $R$ for $F$ and apply those rules to each side of the equation. Such a canonical set is generated by the Knuth-Bendix completion procedure ( [**32**], [**29**], [**13**], [**7**]). Since the Knuth-Bendix procedure is sound and complete [**27**], both sides have the same normal form if and only if $F \models f$. We thus call this rewrite-based method *proof by normalization*.

Define

$$F_{\mathrm{KB}\rightarrow} = \{(l = r) \in \mathcal{E} \mid l \longrightarrow_R^* s \longleftarrow_R^* r\}.$$

The Knuth-Bendix algorithm is presented below.

### 3.2.2. Basic Knuth-Bendix Procedure

It will be useful to view completion as a function which takes a set of equations and inequations and returns a set of rewrite rules and inequations. Thus we define KB : $\mathcal{P}(\mathcal{E}_\Sigma \cup \neg\mathcal{E}_\Sigma) \to \mathcal{P}(\mathcal{R}_\Sigma \cup \neg\mathcal{E}_\Sigma)$ to be the Knuth-Bendix algorithm. For a system $F \subseteq \mathcal{F}_\Sigma$, the algorithm begins in the initial state $(E_0, U_0, \phi)$, where $E_0 = F \cap \mathcal{E}_\Sigma$ and $U_0 = F \cap \neg\mathcal{E}_\Sigma$. The following inference rules (see [**13**] and [**26**] for these with equations alone) are then repeatedly applied in order:

| | | | | |
|---|---|---|---|---|
| *Delete* | $(E \cup \{s = s\}, U, R)$ | $\Rightarrow$ | $(E, U, R)$ | |
| *Compose* | $(E, U, R \cup \{s \to t\})$ | $\Rightarrow$ | $(E, U, R \cup \{s \to u\})$ | if $t \to_R u$ |
| *Simplify* | $(E \cup \{s = t\}, U, R)$ | $\Rightarrow$ | $(E \cup \{s = u\}, U, R)$ | if $t \to_R u$ |
| *Orient* | $(E \cup \{s = t\}, U, R)$ | $\Rightarrow$ | $(E, U, R \cup \{s \to t\})$ | if $s > t$ |
| *Collapse* | $(E, U, R \cup \{s \to t\})$ | $\Rightarrow$ | $(E \cup \{u = t\}, U, R)$ | if $s \to_R u$ |
| *Deduce* | $(E, U, R)$ | $\Rightarrow$ | $(E \cup \{s = t\}, U, R)$ | if $s \leftarrow_R u \to_R t$ |

If this process terminates in $N$ steps with final state $(E_N, U_N, R_N)$ then we say that $\mathrm{KB}(F) = U_N \cup R_N$. ($E_N$ will always be empty if the procedure terminates). If the process does not terminate then we say that the canonical system for $F$ is *unbounded*.

It is also possible for the process to fail if at some stage the rewrite rule system $R_k$ is non-terminating. This cannot happen when using a total-degree ordering but may happen for other orderings (see Example 3.2). Dauchet and Tison [**12**] give an algorithm for deciding the termination of a rewrite system, and this can be used if necessary to check for failure. If the procedure does not fail in this way and terminates we say it is *successful*.

Note that the inequations in $F$ are unaffected by the KB procedure. That is $U_0 = U_N$, or alternatively

$$\mathrm{KB}(F) = \mathrm{KB}(E_0) \cup U_0.$$

We will later extend the list of inference rules given above to process inequations as well.

EXAMPLE 3.1 (Normalization proof). The majority of examples in this chapter use a toy world described by the signature

$$\Sigma = \left\{ \begin{array}{l} \mathsf{John, Jill, Alice, Lucia, George, Bob, Paul} \in \mathsf{person}, \\ \mathsf{father, mother, husband, wife}, \\ \mathsf{sister, fatherinlaw} : \mathsf{person} \to \mathsf{person}, \\ \mathsf{male} : \mathsf{person} \to \mathsf{sentence} \end{array} \right\}.$$

We use this signature to represent knowledge about the family tree given in Figure 3.1.

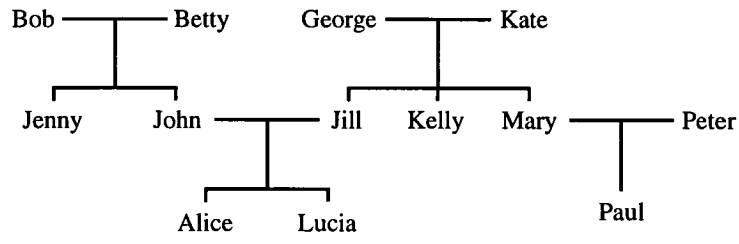



FIGURE 3.1. Motivating family tree

Consider the theorem

$$F = \{\mathsf{father\ sister} = \mathsf{father}, \mathsf{sister\ Alice} = \mathsf{Lucia}\},$$
$$f = (\mathsf{father\ Alice} = \mathsf{father\ Lucia}).$$

In proving this by normalization we first apply KB to $F$ to obtain the rewrite system

$$R = \left\{ \begin{array}{l} \mathsf{father\ sister} \rightarrow \mathsf{father},\ \mathsf{sister\ Alice} \rightarrow \mathsf{Lucia}, \\ \mathsf{father\ Alice} \rightarrow \mathsf{father\ Lucia} \end{array} \right\}.$$

The normal form of the left-hand side of $f$ with respect to $R$ is father Lucia, as is that of the right-hand side, thus proving the theorem. $\diamond$

### 3.2.3. Uniqueness of Canonical Rewrite Systems

DEFINITION 3.3. A rewrite system $R$ is *convergent* for a set of equations $F$ if $R$ presents the same theory as does $F$, i.e. if $s \leftrightarrow_R^* t$ if and only if $s =_F t$.

DEFINITION 3.4. A rewrite system $R$ is *reduced* if for all $(s \rightarrow t) \in R$, $t$ is in normal form with respect to $R$ and $s$ is in normal form with respect to $R \setminus (s \rightarrow t)$.

DEFINITION 3.5. A reduced and convergent rewrite system is said to be *canonical*.

Huet [**27**] has shown that the output of KB($F$) is a canonical system for $F$. If we do not have a total ordering on our terms then an equational theory may have more than once canonical rewrite system, as seen in [**13**]. If we allowed variables in the algebra then obtaining a total ordering may be difficult, and even if we had one the best we can say is that the canonical rewrite systems will be isomorphic [**14**]. By working with our ground algebra we can always obtain a total ordering, as given in Section 2.3.4, and furthermore we have the following strong result:

THEOREM 3.1. *A set of equations $F$, with fixed term ordering, has a unique canonical rewrite system.*

PROOF. Let $R_1, R_2$ be two canonical rewrite systems for $F$, with $R_1 \neq R_2$. Consider any $(l \to r) \in R_2 \setminus R_1$. Since $R_2$ is convergent we have that $l =_F r$, so we must have a rewrite proof of $l =_F r$ from the rules in $R_1$, as $R_1$ is also convergent for $F$.

Suppose $r$ is not a normal form under $R_1$. Then we have $l \to^*_{R_1} s \leftarrow^*_{R_1} r$, for some $s$, with $l > r > s$. Again, since $R_1$ is convergent for $F$, we have $r =_F s$, so in $R_2$ we must have $r \to^*_{R_2} t \leftarrow^*_{R_2} s$, for some $t$, with $r > s \geq t$. This gives $l \to^*_{R_2} t$, with $r > t$, which contradicts $R_2$ being reduced.

If $r$ is already a normal form under $R_1$ then we must have $l \to^*_{R_1} s$, since $l \not\equiv r$, with $l > s$. Thus $l =_F s$ so under $R_2$ we have $l \to^*_{R_2} t \leftarrow^*_{R_2} s$, for some $t$, with $l > s \geq t$. This now gives $l \to^*_{R_2} t$, with $l > t$, which again contradicts $R_2$ being reduced.

Either case gives a contradiction, so we must have $R_1 = R_2$. □

This theorem is motivated by the corresponding result for reduced Gröbner bases [**9**]. Polynomial algebras share the lack of variables with our ground term algebras, making the completion procedures for each very similar. This is particularly apparent when looking at the non-commutative polynomials of [**34**].

The uniqueness theorem gives rise to several other results which are useful in formulating proof strategies.

COROLLARY 3.1. *For a given input $F$ and fixed term ordering, the output of KB is independent of the order in which the equations in $F$ are processed.*

PROOF. As mentioned above, the output of KB is a canonical rewrite system for $F$. Theorem 3.1 gives that this is then necessarily independent of equation ordering. □

This trivial result is not so trivial when extending KB to process inequations. We include it here for comparison with the development of KBC in Section 3.4.4.

In our applications we will almost exclusively use a total degree ordering on our terms (see Section 2.3.4). This guarantees that at every stage of the Knuth-Bendix process the set of rewrite rules $R$ is terminating. However, sometimes a lexicographic ordering is more appropriate for a particular problem, as in the following example:

EXAMPLE 3.2 (Lexicographic ordering). Consider a model of the group of square symmetries with two generators h and r. We view these as a reflection and a rotation which map the corner points of a square onto its corner points, and so give a signature

$$\Sigma_D = \{\ \mathsf{h}, \mathsf{r} : \mathsf{point} \to \mathsf{point}\ \}.$$

(This model is presented in more detail in Section 6.4.2). Three equations are sufficient to define the behaviour of the two functions, giving

$$F_D = \{\ \mathsf{h\ h = i},\ \mathsf{r\ r\ r\ r = i},\ \mathsf{r\ h = h\ r\ r\ r}\ \}.$$

With a total-degree ordering on terms, if $r > h$ we obtain the canonical system

$$R_1 = \{\ \mathsf{h\ h \to i},\ \mathsf{r\ h\ r \to h},\ \mathsf{r\ r\ r \to h\ r\ h},\ \mathsf{r\ r\ h \to h\ r\ r}\ \}$$

while if $h > r$ then we get the larger system

$$R_2 = \left\{ \begin{array}{l} \mathsf{h\ h \to i},\ \mathsf{r\ r\ r\ r \to i},\ \mathsf{h\ r\ h \to r\ r\ r}, \\ \mathsf{h\ r\ r \to r\ r\ h},\ \mathsf{r\ h\ r \to h},\ \mathsf{r\ r\ r\ h \to h\ r} \end{array} \right\}.$$

The real aim of finding a canonical system for $F_D$ is to be able to determine which sequences of rotations and reflections are equivalent by finding normal forms for them. The normal forms produced by $R_1$ and $R_2$ can be rather strange in relation to the original model. The natural and pragmatic normal form would instead be a sequence of reflections followed by a series of rotations, and this is what we obtain with a lexicographic ordering and $r > h$:

$$R_3 = \{\ \mathsf{h\ h \to i},\ \mathsf{r\ r\ r\ r \to i},\ \mathsf{r\ h \to h\ r\ r\ r}\ \}.$$

Even though the last rule increases the length of a term, we know it is terminating because it moves an h to the left at every step.

Unfortunately, lexicographic ordering is not perfect for this example, since if we chose $h > r$ then during KB *Deduce* would produce a new equation $\mathsf{r\ h\ r = h}$ which is oriented as $\mathsf{h \to r\ h\ r}$ and gives a non-terminating system. Hence for this ordering the KB procedure fails. Since this can be detected, as in [14], some form of backtracking may be useful. ◇

Given then that we may be interested in using lexicographic orderings, we rephrase the previous corollary as:

COROLLARY 3.2. *For a given input $F$ and fixed term ordering, if KB succeeds then the output $KB(F)$ is unique.*

The last result we are interested in gives an important relationship between equivalent equational systems and KB. Constructing an algorithm for systems with inequations with the same relationship will be our focus in Section 3.4.4.

DEFINITION 3.6. If $F_1 \models e$ if and only if $F_2 \models e$ then $F_1$ and $F_2$ are said to be *equivalent systems of equations.*

COROLLARY 3.3. *Two systems of equations $F_1$ and $F_2$ are equivalent if and only if $KB(F_1) = KB(F_2)$.*

PROOF. If $F_1$ and $F_2$ are equivalent then $s =_{F_1} t$ if and only if $s =_{F_2} t$. We can modify the proof of Theorem 3.1, using $F_1, F_2$ for $R_1, R_2$, to show that $\mathrm{KB}(F_1) = \mathrm{KB}(F_2)$.

The converse holds by the soundness of KB. □

This corollary justifies proof by invariance, to be discussed in the following section. Also, since $F$ and $R = \mathrm{KB}(F)$ are equivalent, it gives the additional result:

COROLLARY 3.4. *For a set of equations $F$ and an equation $f$,*

$$KB(F \cup f) = KB(KB(F) \cup f).$$

### 3.2.4. Proof by Invariance

A second method of determining whether $f \in \mathcal{F}$ is entailed by $F \subseteq \mathcal{E}$ is to apply the Knuth-Bendix algorithm KB to both the sets $F$ and $F \cup f$. If $f$ is indeed implied by $F$ then $F$ and $F \cup f$ are equivalent, so by Corollary 3.3 the results of KB will be the same.

We write

$$F \models_{\mathrm{KB}} f \quad \text{if} \quad \mathrm{KB}(F) = \mathrm{KB}(F \cup f),$$

and define $F_{\mathrm{KB}} = \{f \in \mathcal{F} \mid F \models_{\mathrm{KB}} f\}$.

PROPOSITION 3.1. $F_{KB_{\rightarrow}} = F_{KB}$.

PROOF. If $f \in F_{\mathrm{KB}_{\rightarrow}}$ then by the soundness and completeness of the Knuth-Bendix procedure we have that $F \models f$. Thus the two sets of equations $F$ and $F \cup f$ are equivalent, and hence by Corollary 3.3 we have $\mathrm{KB}(F) = \mathrm{KB}(F \cup f)$, so that $f \in F_{\mathrm{KB}}$.

Conversely, if $\mathrm{KB}(F) = \mathrm{KB}(F \cup f)$ then at some stage in the right-hand application of KB the inference rule *Delete* must have been used to remove $f$. That is the equations of $F$ must be used to reduce both sides of $f$ to the same term, so that $f \in F_{\mathrm{KB}_{\rightarrow}}$. □

EXAMPLE 3.3 (Invariance proof). Consider again the theorem given in Example 3.1. The canonical system for the hypotheses $F$ is the given

$$R = \left\{ \begin{array}{l} \mathsf{father\ sister} \rightarrow \mathsf{father},\ \mathsf{sister\ Alice} \rightarrow \mathsf{Lucia}, \\ \mathsf{father\ Alice} \rightarrow \mathsf{father\ Lucia} \end{array} \right\}.$$

Depending on the order in which KB processes the equations, when finding the canonical system for $F \cup f$ either $f$ will be reduced to a tautology by the generated rule $\mathsf{father\ Alice} \rightarrow \mathsf{father\ Lucia}$ or the same generated rule will be itself reduced by the oriented conclusion. Hence $\mathrm{KB}(F) = \mathrm{KB}(F \cup f)$ so that $F \models f$. ◇

Note that from the earlier identity $\mathrm{KB}(F) = \mathrm{KB}(E_0) \cup U_0$ we can never use $\models_{\mathrm{KB}}$ to prove that an inequation is the consequence of a system.

### 3.2.5. Proof by Contradiction

An alternative method of proving an equation to be the consequence of a system is proof by *contradiction.* We will show that this method can prove any equation that could be proved by normalization or invariance. Additionally proof of inequations from a system of equations and inequations is immediately possible, information about non-theorems can be obtained from the result of the contradiction method, and a contradiction proof incorporates a useful semidecidability (as seen in Example 3.5).

To prove the theorem $F \models f$ by contradiction we find the canonical form of the system $F \cup \neg f$. If it includes a contradiction then the theorem is true, otherwise it is false.

To find contradictions in the system we need to extend the basic Knuth-Bendix procedure so that inequations are processed. For the system $F = E_0 \cup U_0$, this can simply be done by completing $E_0$ alone and reducing the inequations in $U_0$ with the result. However, this separation is somewhat artificial in that we would like to view $F$ as a complete theory in itself. This is highlighted by a case where $\mathrm{KB}(E_0)$ is unbounded but $E_0 \cup U_0$ is inconsistent. Contradiction then gives a semidecidable proof procedure, as illustrated in Example 3.5.

To define a proof by contradiction method we extend the set of inference rules for KB to obtain a new procedure, Knuth-Bendix with *inequations*, or KBI, which we can apply to the whole theory $F$. The two rules needed are the following:

$$\begin{array}{lllll} \textit{Contradiction} & (E \cup \{s \neq s\}, R) & \Rightarrow & \bot & \\ \textit{SimplifyU} & (E \cup \{s \neq t\}, R) & \Rightarrow & (E \cup \{s \neq u\}, R) & \text{if } t \rightarrow_R u \end{array}$$

These two inference rules generate no additional equations or inequations and so do not effect the termination of the Knuth-Bendix procedure. Specifically, if KB had terminated then the equation set given as input would have been reduced to the empty set. Unless a contradiction is obtained, the equation set in KBI will also reduce to the empty set. The number of inequations will be the same, but each will be in reduced form with respect to the generated rewrite rules.

When trying to prove a non-theorem by contradiction these reduced inequations and the rewrite rules are useful in finding additional hypotheses which will give a theorem. The generating of such *side conditions* is discussed in Section 3.5.

Since the completion of the equations in a system $F$ is unaffected by the inequations in $F$, we have the following simple result:

LEMMA 3.1. *For $F \subseteq \mathcal{F}$ with $KBI(F) \neq \bot$,*

$$KBI(F) \cap \mathcal{R} = KB(F \cap \mathcal{E}).$$

With KBI : $\mathcal{P}(\mathcal{F}) \to \mathcal{P}(\mathcal{R} \cup \neg\mathcal{E})$ we can introduce the corresponding notion of entailment. We define $\models_{\text{KBI}_\neg}$ by

$$F \models_{\text{KBI}_\neg} f \quad \text{if} \quad \text{KBI}(F \cup \neg f) = \bot,$$

and again put $F_{\text{KBI}_\neg} = \{f \in \mathcal{F} \mid F \models_{\text{KBI}_\neg} f\}$.

PROPOSITION 3.2. $F_{KB} = F_{KBI_\neg}$.

PROOF. Suppose $(l = r) \in F_{\text{KB}}$ so that $\text{KB}(F \cup (l = r)) = \text{KB}(F)$. Then at some stage of KB the inference rule *Delete* must be invoked, so that $(l = r)$ must reduce to give an equation $(s = s)$. But then in the application of KBI the corresponding inequation $(l \neq r)$ will produce the inequation $(s \neq s)$, a contradiction, so that $\text{KBI}(F \cup (l \neq r)) = \bot$, giving $(l = r) \in F_{\text{KBI}_\neg}$.

The proof of the converse is similar. □

This result shows that we can prove anything that could be proved using KB alone. Additionally we are now able to prove theorems with hypotheses or conclusion in $\neg\mathcal{E}$, as discussed in Section 3.4.1.

We can now be more precise about the notions of *theorem* and *proof*. For $F \subseteq \mathcal{F}$ and $f \in \mathcal{F}$ we call the pair $(F, f)$ a *possible theorem*. If $F \models_{\text{KBI}_\neg} f$ then we call $(F, f)$ a *theorem* and write $F \Rightarrow f$. If $\text{KBI}(F \cup \neg f) \neq \bot$ we call $(F, f)$ a *non-theorem* and write $F \not\Rightarrow f$. We will talk about *proving* a possible theorem $(F, f)$, the empirical act of computing $\text{KBI}(F \cup \neg f)$, and say that $(F, f)$ is *proved* if we find that $F \Rightarrow f$.

EXAMPLE 3.4 (Contradiction proof). Returning again to Example 3.1, we want to prove the possible theorem $(F, f)$, where

$$F = \{\mathsf{father\ sister} = \mathsf{father},\ \mathsf{sister\ Alice} = \mathsf{Lucia}\},$$
$$f = (\mathsf{father\ Alice} = \mathsf{father\ Lucia}).$$

We thus apply KBI to the augmented system

$$F \cup \neg f = \left\{ \begin{array}{l} \mathsf{father\ sister} = \mathsf{father}, \mathsf{sister\ Alice} = \mathsf{Lucia}, \\ \mathsf{father\ Alice} \neq \mathsf{father\ Lucia} \end{array} \right\}.$$

Here the first two equations are used to generate the rule $\mathsf{father\ Alice} \to \mathsf{father\ Lucia}$ which then reduces the single inequation to $\bot$. Hence $(F, f)$ is a theorem, that is $F \Rightarrow f$. ◇

EXAMPLE 3.5 (Unbounded hypotheses). Consider the set of equations

$$F = \left\{ \begin{array}{l} \mathsf{father\ Jill} = \mathsf{George}, \\ \mathsf{fatherinlaw\ father} = \mathsf{father\ mother} \end{array} \right\},$$

from which we would like to prove the conclusion

$$\mathsf{father\ mother\ Jill} = \mathsf{fatherinlaw\ George}.$$

A normalization proof requires a canonical system for $F$ but $\mathrm{KB}(F)$ is unbounded, containing the rules

$$\mathsf{father\ mother}^k\ \mathsf{Jill} \to \mathsf{fatherinlaw}^k\ \mathsf{George}$$

for all $k \geq 1$. Hence the first part of the normalization proof cannot succeed.

For a contradiction proof the negated conclusion leads to $\perp$ with the first rule generated, thus terminating KBI and proving the theorem. $\diamond$

In an implementation of KBI we must naturally try to detect an unbounded completion process and terminate the procedure. Since the rewrite system is maintained in reduced form one way of doing this is to set an arbitrary bound on the length of terms that can be generated, declaring a system to be unbounded if we ever produce a new rule involving a term of length greater than this bound. If we abort in this way as part of a proof by contradiction then we will assume that we have a non-theorem.

EXAMPLE 3.6 (Non-theorem). Consider now

$$F = \{\mathsf{father\ sister} = \mathsf{father},\ \mathsf{sister\ Alice} = \mathsf{Lucia}\},$$
$$f = (\mathsf{father\ Alice} = \mathsf{John}).$$

Applying KBI to $F \cup \neg f$, with $\mathsf{Alice} > \mathsf{Lucia}$, gives

$$\left\{ \begin{array}{l} \mathsf{father\ sister} \to \mathsf{father},\ \mathsf{sister\ Alice} \to \mathsf{Lucia}, \\ \mathsf{father\ Alice} \to \mathsf{father\ Lucia},\ \mathsf{father\ Lucia} \neq \mathsf{John} \end{array} \right\}.$$

Thus $F$ and $f$ do not constitute a theorem since a contradiction was not found. However, the result of KBI can be used to determine what conditions need to be added to the hypotheses $F$ to give a theorem. This process is considered in Section 3.5. $\diamond$

Working with a variable-free algebra means we have a trivial semantics for our systems, reflected in the simple definition of consistency (Definition 3.2). This allows the following important result, giving a test for consistency.

LEMMA 3.2. *$F \subseteq \mathcal{F}$ is consistent if and only if $KBI(F) \neq \perp$.*

PROOF. Let $E = F \cap \mathcal{E}$, the equations in $F$. By Lemma 3.1 the rules, $R$, generated for $E$ by KBI are a canonical system for $E$. Thus if $F$ is not consistent then by the completeness of $R$ there is a rewrite reduction of some $(s \neq t) \in F$ to a contradiction, so that $\mathrm{KBI}(F) = \perp$. Similarly, if $\mathrm{KBI}(F)$ returns $\perp$ then by the soundness of $R$ there is an equational proof $s =_E t$ for some $(s \neq t) \in F$, so that $F$ is inconsistent. $\square$

### 3.2.6. Proving Compound Sentences - Conjunction and Disjunction

So far we have only looked at proving single equations, but the process can be extended to general sentences. The two simplest cases are conclusions involving the external connectives of conjunction ($\wedge$) and disjunction ($\vee$). Firstly, to prove a possible theorem $(F, f_1 \wedge \cdots \wedge f_n)$ we must try proving each $(F, f_1), \ldots, (F, f_n)$ in turn, obtaining a theorem only if each subproof was successful (i.e. finding $\mathrm{KBI}(F \cup \neg f_j) = \bot$ for each $j$).

EXAMPLE 3.7 (Conjunctive conclusion). Consider the possible theorem with

$$F = \left\{ \begin{array}{l} \mathsf{male\ father} = \mathsf{True}\ !,\ \mathsf{father\ sister} = \mathsf{father}, \\ \mathsf{sister\ Alice} = \mathsf{Lucia},\ \mathsf{father\ Lucia} = \mathsf{John} \end{array} \right\},$$

$$f = (\mathsf{male\ John} = \mathsf{True}) \wedge (\mathsf{father\ Alice} = \mathsf{John}).$$

We find $\mathrm{KBI}(F \cup \{\mathsf{male\ John} \neq \mathsf{True}\}) = \bot$ and $\mathrm{KBI}(F \cup \{\mathsf{father\ Alice} \neq \mathsf{John}\}) = \bot$, so that $F \Rightarrow f$. $\diamond$

The case where the conclusion involves a disjunction, and indeed when it is any compound sentence in general, can be handled in a similar manner. A proof is attempted for each atomic sentence and then the results are combined to determine whether the whole compound sentence is a consequence of the hypotheses. For instance, to prove a possible theorem $(F, f_1 \vee \cdots \vee f_n)$ we must prove at least one of $(F, f_1), \ldots, (F, f_n)$, requiring possibly $n$ separate applications of KBI.

However, for disjunction we have an alternative approach which allows proof in just one step by noting that $\neg(f_1 \vee \cdots \vee f_n)$ is equivalent to $(\neg f_1 \wedge \cdots \wedge \neg f_n)$. Thus the negation of our conclusion is simply a conjunction of inequations, a *set* of inequations in the same way the set of hypotheses represents a conjunction. We can then apply KBI to $F \cup \{\neg f_1, \ldots, \neg f_n\}$ to determine whether $(F, f_1 \vee \cdots \vee f_n)$ is a theorem.

EXAMPLE 3.8 (Disjunctive conclusion). To prove the possible theorem with

$$F = \left\{ \begin{array}{l} \mathsf{male\ father} = \mathsf{True}\ !,\ \mathsf{father\ sister} = \mathsf{father}, \\ \mathsf{father\ Lucia} = \mathsf{John} \end{array} \right\},$$

$$f = (\mathsf{male\ John} = \mathsf{True}) \vee (\mathsf{father\ Alice} = \mathsf{John}).$$

we apply KBI to

$$F \cup \{\mathsf{male\ John} \neq \mathsf{True}, \mathsf{father\ Alice} \neq \mathsf{John}\}.$$

A contradiction is obtained since the hypotheses reduce male John to True, giving $F \Rightarrow f$. $\diamond$

It is easy to see algorithmically why this works. To prove the disjunction we need to show at least one of the negated equations gives a contradiction. By adding all of the negated equations to the hypotheses then if there is at least one contradiction the whole system will reduce to $\bot$.

With this simple method for handling disjunctions we can work with any conclusion by expressing it in *conjunctive normal form.*

## 3.3. Proving Inductive Theorems

We begin this section with an example of a theorem that none of the preceding methods can prove.

EXAMPLE 3.9 (Non-deductive theorem). Consider the signature

$$\Sigma = \left\{ \begin{array}{l} \mathsf{True}, \mathsf{False} \in \mathsf{sentence}, \\ \mathsf{not} : \mathsf{sentence} \longrightarrow \mathsf{sentence} \end{array} \right\}$$

and the possible theorem with

$$F = \{\mathsf{not\ False} = \mathsf{True},\ \mathsf{not\ True} = \mathsf{False}\},$$
$$f = (\mathsf{not\ not} = \mathsf{i}).$$

The reason that we might think of $(F, f)$ as a theorem is that we have complete information about the behaviour of **not** and know that the conclusion holds for all entities in its domain. Yet $F$ has the simple canonical system

$$R = \{\mathsf{not\ False} \longrightarrow \mathsf{True}, \mathsf{not\ True} \longrightarrow \mathsf{False}\},$$

which clearly cannot reduce $\neg f$ to a contradiction. Hence $F \not\vdash f$. $\diamond$

The theorems we have looked at so far and have proved have all been *deductive* in nature. Our methods fail on this example as it is an *inductive* theorem. We will look extensively at the generation of such theorems in Chapter 5, but for now we give some definitions which motivate a corresponding proof method.

### 3.3.1. Strong Completeness

Let $\mathcal{F} = \mathcal{E} \cup \neg\mathcal{E}$, the set of all equations and inequations between terms in $T_\Sigma$. So far we have only defined $s =_E t$ for equations $E$; for $F \subseteq \mathcal{F}$ we write $s =_F t$ if $s =_{F \cap \mathcal{E}} t$.

DEFINITION 3.7. Let $F \subseteq \mathcal{F}$. A rewrite system $R$ is *weakly complete* for $F$ if whenever $s =_F t$ then $s \leftrightarrow_R t$.

This is the sense in which the result of Knuth-Bendix completion is complete, and (along with soundness) is the justification of our deductive proof methods. However, an alternative notion of completeness can be given in terms of consistency (see, for example, [**6**] and [**30**]).

DEFINITION 3.8. A system $F \subseteq \mathcal{F}$ is *strongly*, or *absolutely complete* if for any $e \in \mathcal{E}$ that is not a theorem of $F$ then $F \cup e$ is inconsistent.

Hence if we have a strongly complete system then Lemma 3.2 gives a straightforward procedure for deciding whether a particular equation is a theorem of that system or not. This is the technique of proof by *consistency*, also known as "inductionless induction" because it allows the proof of inductive theorems without the use of standard structural induction [**23**], [**31**]. To apply proof by consistency we need first to define more precisely what we mean by a theorem and then how we may characterize strongly complete systems. The following discussion develops the ideas in [**31**] for our proof system.

DEFINITION 3.9. For $F \subseteq \mathcal{F}$ and $e \in \mathcal{E}$, if $\mathrm{KBI}(F \cup \neg e) = \bot$ then we call $e$ a *deductive theorem* of $F$ and write (as before) $F \Rightarrow e$.

All of the theorems we have proved in previous sections have been deductive theorems, while the theorem of Example 3.9 was not. We write $\mathcal{D}_F$ for the set of all deductive theorems of $F$. Note that if $F$ is inconsistent then $\mathcal{D}_F = \mathcal{E}$. We say that $e \in \mathcal{E}$ is *consistent with* $F$ if $F \cup e$ is consistent, and write $\mathcal{C}_F$ for the set of all equations that are consistent with $F$.

DEFINITION 3.10. $C \subseteq \mathcal{E}$ is a *consistent theory* of $F$ if $C$ is consistent and $\mathcal{D}_F \subseteq C$ (and hence $F \subseteq C$). $C$ is a *maximally consistent theory* if it is a consistent theory of $F$ and there is no $e \in \mathcal{E}$ such that $C \cup e$ is consistent.

Thus each consistent theory $C$ of $F$ satisfies $\mathcal{D}_F \subseteq C \subseteq \mathcal{C}_F$. The following lemma shows that $\mathcal{D}_F$ is itself a consistent theory for $F$ and is thus the smallest consistent theory.

LEMMA 3.3. *If $F \subseteq \mathcal{F}$ is consistent then $\mathcal{D}_F$ is a consistent theory for $F$.*

PROOF. Since $F$ is consistent, $\mathrm{KBI}(F) \neq \bot$. Let $E = F \cap \mathcal{E}$ so that $R = \mathrm{KBI}(F) \cap \mathcal{R} = \mathrm{KB}(E)$. Since $E \Rightarrow f$ for each $f \in \mathcal{D}_F$, we can perform a series of proofs by invariance to find that $\mathrm{KB}(E \cup \mathcal{D}_F) = R$. But $F$ is consistent so we know $R$ does not reduce any inequation in $F$ to a contradiction, and hence $\mathcal{D}_F$ is a consistent theory. □

Furthermore, we can similarly show that if $e \in \mathcal{E}$ is consistent with $F$ then it is also consistent with $\mathcal{D}_F$.

EXAMPLE 3.10 (Unique maximal theories). For the $F$ of Example 3.9, the deductive theorems of $F$ are

$$\mathcal{D}_F = \bigcup_{i,j \geq 0} \{\mathsf{not}^{2i}\ \mathsf{False} = \mathsf{not}^{2j}\ \mathsf{False},\ \mathsf{not}^{2i}\ \mathsf{False} = \mathsf{not}^{2j+1}\ \mathsf{True}\},$$

where we write $\mathsf{not}^0$ for i. The set

$$C_1 = \mathcal{D}_F \cup \{\mathsf{not\ not\ not} = \mathsf{not}\}$$

is a consistent theory for $F$ while

$$C_1' = \mathcal{D}_F \cup \bigcup_{i,j \geq 0} \{\mathsf{not}^{2i} = \mathsf{not}^{2j},\ \mathsf{not}^{2i+1} = \mathsf{not}^{2j+1}\}$$

is a maximally consistent theory for $F$. It is straightforward to see that $C_1'$ is actually unique as a maximal theory. $\diamond$

EXAMPLE 3.11 (Multiple maximal theories). Consider a smaller system

$$F' = \{\mathsf{not\ False} = \mathsf{True}\},$$

with deductive theorems

$$\mathcal{D}_{F'} = \bigcup_{i \geq 1} \{\mathsf{not}^i\ \mathsf{False} = \mathsf{not}^{i-1}\ \mathsf{True}\}.$$

This time both

$$C_1 = \mathcal{D}_{F'} \cup \{\mathsf{not\ not} = \mathsf{i}\}$$
$$\text{and } C_2 = \mathcal{D}_{F'} \cup \{\mathsf{not\ not} = \mathsf{not}\}$$

are consistent theories for $F$ but $C_1 \cup C_2$ is *not* consistent, since we have

$$\mathsf{False} =_{C_1} \mathsf{not\ not\ False} =_{C_2} \mathsf{not\ False} =_F \mathsf{True}.$$

Thus we cannot have a single maximally consistent theory for $F$ since both $C_1$ and $C_2$ can be extended to a maximal theory. Indeed we find that

$$C_1' = \mathcal{D}_{F'} \cup \bigcup_{i,j \geq 0} \{\mathsf{not}^{2i} = \mathsf{not}^{2j},\ \mathsf{not}^{2i+1} = \mathsf{not}^{2j+1}\}$$
$$\text{and } C_2' = \mathcal{D}_{F'} \cup \bigcup_{i,j \geq 0} \{\mathsf{not}^{2i} = \mathsf{not}^{2j+1}\}$$

are both maximally consistent theories for $F$. $\diamond$

DEFINITION 3.11. $f \in \mathcal{E}$ is a *theorem* of $F \subseteq \mathcal{F}$ if it is in the intersection of all maximally consistent theories of $F$.

We use $\mathcal{T}_F$ to denote the set of all theorems of $F$. Since $\mathcal{D}_F$ is a subset of all consistent theories, all deductive theorems are immediately theorems of $F$.

LEMMA 3.4. *If $F \subseteq \mathcal{F}$ has a unique maximally consistent theory $C$ then $C = \mathcal{C}_F$. Conversely, if there is a consistent theory $C$ such that $C = \mathcal{C}_F$ then $C$ is maximal and unique.*

PROOF. Suppose $C \neq \mathcal{C}_F$ so that there is some $f \in \mathcal{C}_F \setminus C$, since $C \subseteq \mathcal{C}_F$. Then $\mathcal{D}_F \cup f$ is a consistent theory for $F$ and can be extended to a maximally consistent theory $C'$ by adding elements of $\mathcal{C}_F$ until no longer possible. This contradicts the uniqueness of $C$, since $f$ is in $C'$ but not in $C$. The converse holds similarly. $\square$

By the definition of a theorem, if an $F$ has a unique maximally consistent theory $C$ then the theorems of $F$ are exactly the elements of $C$. We follow [**31**] and call such a system *unambiguous*. Conversely, if the theorems of $F$ are exactly the elements of some maximally consistent theory $C$, then $C$ must be unique. Lemma 3.4 then gives the important result:

THEOREM 3.2 (Strong Completeness). *$F$ is unambiguous if and only if*

$$\mathcal{T}_F = \mathcal{C}_F,$$

*that is, if and only if the theorems of $F$ are exactly the equations that are consistent with $F$.*

This statement is equivalent to saying that if $F$ is unambiguous then $\mathcal{C}_F$ is the only consistent theory for $F$.

### 3.3.2. Inductive Theorems

The equations we have called theorems of $F$ are exactly those that must hold in any maximal and consistent theory for $F$. That is, if $f$ is a theorem of $F$ then in any consistent model where the equations and inequations in $F$ hold, so must $f$. If $f$ is a deductive theorem then we can indeed prove this relationship, using proof by contradiction. The remaining, non-deductive, theorems will be called inductive theorems of $F$.

DEFINITION 3.12. For $F \subseteq \mathcal{F}$ if $e \in \mathcal{E}$ is a theorem of $F$ and $e \notin \mathcal{D}_F$ then $e$ is an *inductive theorem* of $F$.

The discussion so far has been quite general. In this section and the next we make more precise statements about the nature of inductive theorems and unambiguous systems.

DEFINITION 3.13. A sort $\sigma$ in a signature $\Sigma$ is *entity rich* if there are at least two distinct $\Sigma$-entities $a, b \in \sigma$. A signature $\Sigma$ is entity rich if each sort in $\Sigma$ is entity rich.

We will assume in the following two sections that we have an entity rich signature. Without it most of the ideas and proofs become trivial. If $f : \sigma \to \tau$ and $\tau$ has only a single entity $b$, then any maximal theory must have $fa = b$ and so ambiguity cannot arise.

For our language of a term algebra we can obtain the following characterization of inductive theorems.

THEOREM 3.3. *If the only inequations in $F \subseteq \mathcal{F}$ are entity inequations and $e \in \mathcal{E}$ is an inductive theorem of $F$, then dom$(e) \neq$ **ground**.*

PROOF. Suppose $e$ is an inductive theorem of $F$ with dom$(e)$ = **ground**. Thus $e$ is of the form $fa = gb$ for some $a, b \in \sigma$, $f, g : \sigma \to \tau$. Then for

each $a', b' \in \Sigma_\tau$ with $a' \not\equiv b'$ we can construct a maximally consistent theory containing $fa = a'$ and $gb = b'$, which hence cannot contain $fa = gb$. But by assumption $fa = gb$ is in the intersection of all maximally consistent theories of $F$ and so for each such $a', b'$ one or both of $fa = a'$ and $gb = b'$ must be inconsistent with $F$. This can only happen if either $fa =_F a''$ for some $a'' \not\equiv a'$ or $gb =_F b''$ for some $b'' \not\equiv b'$, with such an equation needed for each pair $a', b'$.

Since $F$ is consistent, we can have only one datum $fa = c$ in $F$ which cannot exclude the constructed theories which already had $fa = c$. Thus $F$ must also contain some $gb = b''$, but we know that $fa = gb$ is consistent with $F$ so we must have $gb =_F c$. Hence we have

$$fa =_F c =_F gb$$

so that $e \in \mathcal{D}_F$, contradicting the inductive nature of $e$. □

The requirement that $F$ only contain entity inequations is needed to limit to equations the conditions which prevent the contradictory theories in the above proof being constructed. By only admitting equations we were able to show $fa =_F gb$. If we allow general inequations then for each $a'$ we could add to $F$ that $fa \neq a'$, making all of the constructed theories inconsistent with $F$, but not establishing $fa =_F gb$. The additional property we need is the following:

DEFINITION 3.14. Let $F \subseteq \mathcal{F}$ and $f : \sigma \to \tau$. If whenever $fa \neq_F a'$ for all $a' \in \Sigma_\tau \setminus a''$ for some $a'' \in \Sigma_\tau$ then $fa =_F a''$, we say that $F$ is *valuationally consistent*.

For example, if $F$ is valuationally consistent and contains male Alice $\neq$ True then $F$ must also contain male Alice = False. If $F$ is valuationally consistent and $fa \neq_F a'$ for all $a' \in \Sigma_\tau$ then $F$ is inconsistent.

Note that if $F$ has only entity inequations then it is trivially valuationally consistent, so the following result is more general than Theorem 3.3.

THEOREM 3.4. *If $F \subseteq \mathcal{F}$ is valuationally consistent and $e \in \mathcal{E}$ is an inductive theorem of $F$, then $dom(e) \neq$ ground.*

PROOF. Suppose $e$ is an inductive theorem of the form $fa = gb$ for some $a, b \in \sigma, f, g : \sigma \to \tau$. We proceed as in the proof of Theorem 3.3 by noting that for each pair $a', b' \in \Sigma_\tau$ with $a' \not\equiv b'$ we could extend $F$ with $fa = a'$ and $gb = b'$ to give a maximally consistent theory which cannot contain $fa = gb$. This is not possible since $fa = gb$ is an inductive theorem and so for each $a', b'$ either $fa = a'$ or $gb = b'$ must be inconsistent with $F$.

Since $F$ is valuationally consistent we cannot have $fa \neq_F a'$ for all $a' \in \Sigma_\tau$. Suppose there is only one $a''$ for which we don't have $fa \neq_F a''$, so that $fa =_F a''$. This still allows construction of theories with $gb = b'$ for all

$b' \not\equiv a''$. Now if there is only only $b''$ for which we don't have $gb \neq_F b''$ then $gb =_F b''$ and since $fa = gb$ is consistent with $F$ we must have $b'' = a''$. Thus $fa =_F gb$, giving $e \in \mathcal{D}_F$. Similarly, if there is more than one $b'$ for which we don't have $gb \neq_F b'$, then we have theories with $fa = a''$ and each such $gb = b'$. We must thus have $gb =_F b''$ for some $b''$, and again $b'' = a''$ since $fa = gb$ is consistent, giving $e \in \mathcal{D}_F$. A similar argument holds for the case when there is more than one $a'$ for which we don't have $fa \neq_F a'$. □

We will see in the next section that if a system $F$ is unambiguous then it must be *functionally complete*. It is then straightforward to see that this property ensures $F$ is also valuationally consistent.

### 3.3.3. Characterizing Ambiguity

In order to characterize those systems that are unambiguous, and thus allow proof of inductive theorems, we need to look at the entities in the system. As seen in the proofs of the previous section, ambiguity arises when the value of a function applied to an entity in its domain is unspecified. If the codomain of the function is not entity rich then the possible value is limited and ambiguity is not possible. We incorporate this into the following definition.

DEFINITION 3.15. A $\Sigma$-system is *functionally complete* if whenever a $\Sigma$-function $f$ has an entity rich codomain then for any $\Sigma$-entity $a \in \mathrm{dom}(f)$, there is a $\Sigma$-entity $b \in \mathrm{cod}(f)$ such that $fa =_F b$.

THEOREM 3.5. *If a $\Sigma$-system $F$ is unambiguous then $F$ is functionally complete.*

PROOF. Suppose $F$ is not functionally complete. Then there exists a function $f$, with entity rich codomain $\mathrm{cod}(f)$, and an entity $a$ such that $fa$ is not $F$-equivalent to any entity. Since $\mathrm{cod}(f)$ is entity rich we can choose any two $b, c \in \mathrm{cod}(f)$ with each of $fa = b$ and $fa = c$ being consistent with $F$. Hence $F$ must have at least two maximally consistent theories, one with $fa = b$ and one with $fa = c$, contradicting the unambiguousness of $F$. Thus $F$ is functionally complete. □

Of course we are more interested in a sufficient condition for a system to be unambiguous. It turns out that the converse of the above theorem holds if we restrict our attention to systems where the only inequations are the standard entity inequations. That is, if a system $F$ has only entity inequations and is functionally complete, then it is unambiguous. The following example illustrates why this doesn't hold for general inequations.

EXAMPLE 3.12 (Functional completeness with ambiguity). Consider the signature

$$\Sigma = \left\{ \begin{array}{l} \mathsf{John}, \mathsf{Paul} \in \mathsf{person}, \\ \mathsf{parent}, \mathsf{father}, \mathsf{husband} : \mathsf{person} \rightarrow \mathsf{sentence} \end{array} \right\}$$

and system

$$F = \left\{ \begin{array}{l} \mathsf{parent\ John} = \mathsf{True}, \mathsf{parent\ Paul} = \mathsf{True}, \\ \mathsf{father\ John} = \mathsf{True}, \mathsf{father\ Paul} = \mathsf{True}, \\ \mathsf{husband\ John} = \mathsf{True}, \mathsf{husband\ Paul} = \mathsf{True}, \\ \mathsf{parent} \neq \mathsf{father} \end{array} \right\}.$$

Then we can form one maximally consistent theory for $F$ containing $\mathsf{parent} = \mathsf{husband}$ and one with $\mathsf{father} = \mathsf{husband}$. These two theories must be distinct since together we could deduce $\mathsf{parent} = \mathsf{father}$, contradicting the inequation in $F$. Hence $F$ is ambiguous even though it is functionally complete. $\diamond$

The stronger property we need to obtain a sufficient condition is the following:

DEFINITION 3.16. A functionally complete $\Sigma$-system $F$ is *inductively consistent* if whenever $(f \neq g) \in F$, for $\Sigma$-functions $f, g$, there are $\Sigma$-entities $a \in \mathrm{dom}(f)$, $b_1, b_2 \in \mathrm{cod}(f)$, $b_1 \not\equiv b_2$, such that $fa =_F b_1$ and $ga =_F b_2$.

The $F$ in the above example is not inductively consistent. In this case we are unable to add that $\mathsf{parent} \neq \mathsf{father}$ until we can give a person for whom these functions are different.

Note that a system which only has entity inequations is always inductively consistent. The next theorem thus proves and generalizes our comments preceding the above example.

THEOREM 3.6. *If a $\Sigma$-system $F$ is (functionally complete and) inductively consistent, then $F$ is unambiguous.*

PROOF. Suppose $F$ is ambiguous. Thus $F$ has at least two distinct maximally consistent theories $C_1$ and $C_2$. Let $f = g$ be an equation in $C_1 \setminus C_2$. Since $f = g$ is consistent with $F$ and $F$ is functionally complete, for each $a \in \mathrm{dom}(f)$ there is a $b \in \mathrm{cod}(f)$ such that $fa =_F b =_F ga$.

Now $C_2$ is maximal so that the equation $f = g$ must be inconsistent with $C_2$. Thus there is some inequation $h_1 \neq h_2$ in $F$ such that $h_1 =_{C_2 \cup \{f=g\}} h_2$. Since $h_1 = h_2$ cannot be proved from $C_2$ alone, as $C_2$ is consistent, $f = g$ must be involved at least once in the equational proof of $h_1 = h_2$ using a substitution step $kfh = kgh$ for some terms $k$ and $h$. That is, $h_1 =_{C_2 \cup (f=g)} kfh = kgh =_{C_2 \cup (f=g)} h_2$.

But $F$ is inductively consistent so that for some $a \in \mathrm{dom}(h_1)$ and $b_1, b_2 \in \mathrm{cod}(h_1)$, $h_1 a =_F b_1$ and $h_2 a =_F b_2$. Thus in $C_2 \cup (f = g)$ we have $kfha =_{C_2 \cup (f=g)} h_1 a = b_1$ and $kgha =_{C_2 \cup (f=g)} h_1 a = b_1$. Since $F$ is functionally complete there is some $a' \in \mathrm{cod}(h)$ such that $ha =_F a'$, giving $kfa' =_{C_2 \cup (f=g)} b_1$ and $kga' =_{C_2 \cup (f=g)} b_2$. But again by functional completeness, there is some $c \in \mathrm{cod}(f)$ such that $fa' =_F c =_F ga'$ so that $b_1 =_{C_2 \cup (f=g)} kc =_{C_2 \cup (f=g)} b_2$.

We can repeat this argument to show that all uses of $f = g$ in a proof are already implied by $C_2$. Thus $C_2$ is itself inconsistent and so there are no equations in $C_1 \setminus C_2$. Since $C_1$ and $C_2$ were arbitrary, $F$ must have a unique maximally consistent theory and is hence unambiguous. □

EXAMPLE 3.13 (Infinite domain). Suppose we have the single $\Sigma$-function father : person → person and the system $F = \{\text{father Alice} = \text{John}\}$. For $F$ to be functionally complete we must then add an equation for father John, say father John = Bob. But then we need father Bob and so on. In this case functional completeness is unattainable, requiring infinite data for a set of entities which is conceptually finite. ◇

An important implication of this example will arise in Chapter 5. There we will observe that both the father of Alice and the father of John are male and so make the conjecture that male father = True !. But without functional completeness this is a conjecture that can never be proved. Compare this to Example 3.9 where we might similarly make the conjecture from $F$ that not not = i. In that case however we can not only make the conjecture but can also then prove it (by consistency).

This is the situation we have with the natural numbers. We may observe the sequence of the sums of the first $n$ integers

$$1, 3, 6, 10, \ldots$$

and conjecture that the $n$th number in this sequence is $n(n+1)/2$. That is, we conjecture the equation

$$\sum_{j=1}^{n} j = \frac{n(n+1)}{2}.$$

But we are then able to prove our conjecture using mathematical induction. This ability comes down to the structure of the natural numbers, all of which can be built from the successor constructor succ. We are able to define an infinite number of entities succ(0), succ(succ(0)), ..., and, more importantly, can use this structure to completely define the functions of $n$ in the above equation. This gives the functional completeness we need.

We can produce a similar Peano structure for our family tree. In addition to using father as a function we can also use it as an entity constructor, declaring a new entity father(Alice) ∈ person. We can then define the father of Alice to be the entity father(Alice). Continuing the constructing we can declare entities $\text{father}^n(\text{Alice}) \in \text{person}$, for all $n \geq 1$, writing $\text{father}^2(\text{Alice})$ for father(father(Alice)) and so on. The system

$$F = \bigcup_{k \geq 1} \left\{ \begin{array}{l} \text{father father}^k(\text{Alice}) = \text{father}^{k+1}(\text{Alice}), \\ \text{male father}^k(\text{Alice}) = \text{True} \end{array} \right\}$$

is then functionally complete since values of father and male are known for all entities. We can thus prove male father = True ! by consistency.

Of course, this construction is very strong for our modelled world. If the universe is finite then it is meaningless to talk about an infinite sequence of fathers. Both evolutionary theory and religion would say that there is some $K$ such that $\mathsf{father}^K(\mathsf{Alice})$ is no longer an element of **person**. The integers give a signature with a truly infinite number of entities whereas here the number is large but finite.

An alternative approach which covers both cases is to introduce a special element $\omega_\sigma$ for each sort $\sigma$. We use it to indicate a value that is unknowable or meaningless. For instance, we might include equations

$$\mathsf{father\ Adam} = \omega$$

to indicate that we cannot know the father of Adam or

$$\mathsf{wife\ Jill} = \omega$$

to say that it is meaningless to talk about Jill's wife. (This second situation is perhaps better handled by introducing subsorts of **person**).

EXAMPLE 3.14 (Obtaining inductive truth with $\omega$). Consider then the system

$$F = \left\{ \begin{array}{l} \mathsf{father\ Alice} = \mathsf{John},\ \mathsf{male\ John} = \mathsf{True}, \\ \mathsf{male\ Alice} = \mathsf{False},\ \mathsf{father\ John} = \omega \end{array} \right\}.$$

$F$ is then functionally complete and the equation $\mathsf{male\ father} = \mathsf{True}$ ! is an inductive theorem of $F$. $\diamond$

We will say more about the use of $\omega$ at the end of Section 5.1.3.

## 3.4. Proving Inequations

### 3.4.1. Proof by Contradiction

The proof method $\models_{\mathrm{KBI}_\neg}$ presented in Section 3.2.5 extends immediately to theorems where the hypotheses $F$ and conclusion $f$ are drawn from $\mathcal{F}$. So far we have only defined the proof operator $\models$ for equations. Since this is equivalent to proof by contradiction using KBI for equations, we extend $\models$ to inequations by defining

$$F \models f \quad \text{iff} \quad \mathrm{KBI}(F \cup \neg f) = \bot.$$

As before, a possible theorem $(F, f)$ is called a *theorem* if in fact $F \models_{\mathrm{KBI}_\neg} f$. If $(F, f)$ is a theorem we write $F \Rightarrow f$ and otherwise write $F \not\Rightarrow f$ and say that $(F, f)$ is a *non-theorem*. Note that if the hypotheses $F$ are inconsistent then we have a theorem $F \Rightarrow f$ for any $f \in \mathcal{F}$.

EXAMPLE 3.15 (Contradiction proof). Consider the theorem

$$F = \{\mathsf{male\ father} = \mathsf{True}\ !,\ \mathsf{male\ Alice} \neq \mathsf{True}\},$$
$$f = (\mathsf{father\ John} \neq \mathsf{Alice}).$$

In proving this by contradiction, the equation father John = Alice is added to the hypotheses which during KBI gives rise to the critical pair

$$\text{male Alice} \longleftarrow \text{male father John} \longrightarrow \text{True}.$$

The deduced equation then contradicts male Alice $\neq$ True and the theorem is proved. $\diamond$

EXAMPLE 3.16 (Non-theorem). Consider now

$$F = \{\text{male father} = \text{True}\ !\},$$
$$f = (\text{father John} \neq \text{Alice}).$$

Applying KBI to $F \cup \neg f$ gives

$$\left\{ \begin{array}{l} \text{male father} \rightarrow \text{True}\ !,\ \text{father John} \rightarrow \text{Alice}, \\ \text{male Alice} \rightarrow \text{True} \end{array} \right\}.$$

As in Example 3.6, $(F, f)$ is a non-theorem since a contradiction was not found. Again the result of KBI can be used to generate the extra conditions needed for a theorem, as described in Section 3.5. $\diamond$

### 3.4.2. Proving Compound Sentences - Implication

In Section 3.2.6 we saw that we can easily determine whether a disjunction $f_1 \vee \cdots \vee f_n$ is a consequence of an $F$. Now that we have extended our discussion to proving inequations we can use this technique to prove an additional compound form. Recall that the *implication* $f \rightarrow g$ is equivalent to $\neg(f \wedge \neg g)$, in turn equivalent to $\neg f \vee g$. Thus to prove a simple implication we add its antecedent and the negation of its consequent to the hypotheses and look for a contradiction.

EXAMPLE 3.17 (Proving an implication). Consider the possible theorem

$$F = \{\text{father sister} = \text{father},\ \text{father Lucia} = \text{John}\},$$
$$f = (\text{sister Alice} = \text{Lucia}) \rightarrow (\text{father Alice} = \text{John}).$$

Here we want to prove (sister Alice $\neq$ Lucia) $\vee$ (father Alice = John) and so apply KBI to

$$F \cup \{\text{sister Alice} = \text{Lucia}, \text{father Alice} \neq \text{John}\},$$

obtaining $\perp$. $\diamond$

Note that this approach will be applicable if the antecedent of the implication is a conjunction $f_1 \wedge \cdots \wedge f_n$ and the consequent is a disjunction $g_1 \vee \cdots \vee g_m$ since then we apply KBI to

$$F \cup \{f_1, \ldots, f_n, \neg g_1, \ldots, \neg g_m\}.$$

For example, the consequent could be another implication, proving $f \rightarrow (g \rightarrow h)$ by augmenting $F$ with $\{f, g, \neg h\}$. However a disjunctive antecedent or conjunctive consequent will not yield the required form and so must be

approached by expressing the implication in conjunctive normal form and proving each disjunction separately.

As an example of using conjunctive normal form, consider the problem of proving the *equivalence* of two equations. We say two equations $f, g$ are equivalent in a system $F$ if $F \Rightarrow (f \rightarrow g) \wedge (g \rightarrow f)$. This expression reduces to $(\neg f \vee g) \wedge (\neg g \vee f)$ so if we want to prove the possible theorem $(F, f \equiv g)$ then we need to show that both $F \cup \{f, \neg g\}$ and $F \cup \{g, \neg f\}$ reduce to $\bot$.

EXAMPLE 3.18 (Proving an equivalence). Consider an extension of the possible theorem in Example 3.17 where now we want to prove equivalence:

$$F = \{\mathsf{father\ sister} = \mathsf{father},\ \mathsf{father\ Lucia} = \mathsf{John}\},$$
$$f = (\mathsf{sister\ Alice} = \mathsf{Lucia}) \equiv (\mathsf{father\ Alice} = \mathsf{John}).$$

We have already seen one of the necessary subproofs succeed above but we find that the other gives

$$\mathrm{KBI}(F \cup \{\mathsf{father\ Alice} = \mathsf{John}, \mathsf{sister\ Alice} \neq \mathsf{Lucia}\}) \neq \bot,$$

so $F \not\Rightarrow f$. $\diamond$

To conclude this section, note that we also use the symbols $\rightarrow$ and $\equiv$ elsewhere, for rewriting and string equality, respectively. It should be clear from the context when these are instead being used as the implication and equivalence connectives.

### 3.4.3. Compound Hypotheses

In Section 3.2.6 and the previous section we extended the conclusions we could prove using KBI to general sentences of propositional logic. KBI manipulates conjunctions of equations and inequations (as sets) and so by writing a conclusion in conjunctive normal form we break the proof into a number of subproofs, all of which must succeed to prove the theorem. Conjunctive normal form is applicable since the negated disjunctive terms are then conjunctions which can be added the hypotheses for proof by contradiction.

We can similarly extend our hypotheses, but this time we do not negate sentences before adding them and so breaking the problem into subproofs requires *disjunctive* normal form instead. For example, if our hypotheses were of the form $f \wedge (g \vee h)$ then we write them as $(f \wedge g) \vee (f \wedge h)$. To then prove a theorem with hypotheses $\{f, g \vee h\}$ we need to prove the two subtheorems with hypotheses $\{f, g\}$ and $\{f, h\}$.

Note that to prove a theorem with such disjunctive hypotheses we must prove *all* of the corresponding subtheorems, as in the case of a conjunctive conclusion. Since we do not know which parts of the disjunction hold we cannot treat the subtheorems from the disjunction as a disjunction of theorems. That is, being able to prove one subtheorem is not sufficient to prove

the whole theorem. In Section 3.5.2 we will look at what we can say about a non-theorem with disjunctive hypotheses.

Table 3.1 summarizes our extension of proof by contradiction to arbitrary hypotheses and conclusions by giving the disjunctive and conjunctive normal forms of simple sentences involving $\rightarrow$, $\oplus$ (exclusive disjunction), and $\equiv$.

| | DNF | CNF |
|---|---|---|
| $f \rightarrow g$ | $\neg f \vee g$ | $\neg f \vee g$ |
| $f \oplus g$ | $(f \wedge \neg g) \vee (\neg f \wedge g)$ | $(f \vee g) \wedge (\neg f \vee \neg g)$ |
| $f \equiv g$ | $(f \wedge g) \vee (\neg f \wedge \neg g)$ | $(f \vee \neg g) \wedge (\neg f \vee g)$ |

TABLE 3.1. Disjunctive and conjunctive normal forms for sentence connectives $\rightarrow$, $\oplus$, and $\equiv$

EXAMPLE 3.19 (Hypotheses with implication). Consider the simple possible theorem with

$$F = \{(\mathsf{father\ Alice} = \mathsf{John}) \rightarrow (\mathsf{male\ John} = \mathsf{True})\},$$
$$f = (\mathsf{male\ John} = \mathsf{True}).$$

Here we have two subproofs to try, the first with hypotheses $\{\mathsf{father\ Alice} \neq \mathsf{John}\}$ and the second with hypotheses $\{\mathsf{male\ John} = \mathsf{True}\}$. The latter is trivially a theorem but applying KBI to the first gives no reduction, so $F \not\Rightarrow f$. This gives a concrete model of implication since we can never prove the consequent of the implication until either $F \Rightarrow (\mathsf{father\ Alice} = \mathsf{John})$ or $F \Rightarrow (\mathsf{male\ John} = \mathsf{True})$. $\diamond$

Allowing hypotheses with implications gives a similar structure to logic programming [**33**]. We can view our equational language as a first order language without variables and with the single predicate $=$ (actually $=_\sigma$ for each $\Sigma$-sort $\sigma$). The Herbrand universe for this language is $T_\Sigma$ without the internal $=_\sigma$, and the corresponding Herbrand base is $\mathcal{E}_\Sigma$, the set of all equations between terms in $T_\Sigma$ of compatible type. Thus the ground semantics of logic programming are very similar to ours. For example, we can express the single hypothesis above as the simple program

$$P = |(\mathsf{male\ John} = \mathsf{True}) \leftarrow (\mathsf{father\ Alice} = \mathsf{John})\ .$$

The goal $\mathsf{male\ John} = \mathsf{True}$ then requires the subgoal $\mathsf{father\ Alice} = \mathsf{John}$, which is not known, and so proof fails. An analogy of finding *side conditions* (see Section 3.5) in this ground case would be to return all unsatisfied subgoals.

The difference between the methods lies in the syntactic processing carried out during proof by contradiction. Consider a new possible theorem

$$F = \left\{ \begin{array}{l} (\mathsf{male\ Alice} \neq \mathsf{True}) \rightarrow (\mathsf{father\ Paul} \neq \mathsf{Alice}), \\ \mathsf{male\ father\ Paul} \neq \mathsf{True} \end{array} \right\},$$
$$f = (\mathsf{father\ Paul} \neq \mathsf{Alice}).$$

Again we are trying to prove the consequent of an implication and so one of the subproofs will trivially succeed. The second, with hypotheses $\{\mathsf{male\ Alice} = \mathsf{True},\ \mathsf{male\ father\ Paul} \neq \mathsf{True}\}$, also succeeds because the negated conclusion forms a critical pair with the first hypothesis which then reduces the second hypothesis to $\bot$. Thus $F \Rightarrow f$. The corresponding program

$$P = \left| \begin{array}{l} (\mathsf{father\ Paul} \neq \mathsf{Alice}) \leftarrow (\mathsf{male\ Alice} \neq \mathsf{True}) \\ \mathsf{male\ father\ Paul} \neq \mathsf{True} \end{array} \right.$$

is unable to prove the goal $\mathsf{father\ Paul} \neq \mathsf{Alice}$. To achieve this interaction between predicates in logic programming requires the use of variables. We will say more about variables at the end of Section 3.7.

### 3.4.4. Minimal Systems

As observed previously, proof by invariance using KB can never prove an inequation since KB does no inequation processing. The next step would be to extend proof by invariance to a method where KBI is used instead of KB. However, this cannot prove even the trivial theorem

$$(\mathsf{father\ Alice} \neq \mathsf{Paul}) \Rightarrow (\mathsf{father\ Alice} \neq \mathsf{Paul}).$$

The remedy for this particular example is straightforward (namely, extending KBI to remove duplicate inequations) but to obtain a proof method equivalent to proof by contradiction is much harder. This section addresses this problem.

The canonical rewrite system for an equational theory $E$ typically contains many more elements than $E$ itself. A completion procedure needs to add new identities to express information lost when only allowing equations to be used in one direction. For systems with inequations the situation is different. To obtain a contradiction and prove a theorem $F \Rightarrow f$ we need only reduce a single inequation to give $\mathrm{KBI}(F \cup \neg f) = \bot$. It may then be that some inequations in $F$ are redundant for this task, giving a set of inequations $U \subseteq F$ such that $(F \setminus U) \Rightarrow f$ if and only if $F \Rightarrow f$.

We have previously defined two equational theories $E_1$ and $E_2$ to be *equivalent* if $E_1 \models e$ if and only if $E_2 \models e$. By Proposition 3.2, $E \models e$ if and only if $E \Rightarrow e$, and so we now extend our definition and say that $F_1, F_2 \subseteq \mathcal{F}$ are *equivalent* if $F_1 \Rightarrow f$ if and only if $F_2 \Rightarrow f$.

Observe that if $g \in F$ and $F \setminus g \Rightarrow g$ then $F$ and $F \setminus g$ are equivalent. (This is in fact a simple result of $\Rightarrow$ being a consequence relation, as discussed in Section 4.1.2). We say that $F$ is *minimal* (for inequations) if there is no such inequation $g \in F$ such that $F \setminus g \Rightarrow g$.

We can obtain a system with minimal inequations from a system $F$ by applying the following rule:

> for each inequation $f \in F$,
> if $F \setminus f \Rightarrow f$ then remove $f$ from $F$.

Since equations do not interact in any way during proof by contradiction, this process of looking at individual inequations is guaranteed to produce a minimal system and need only examine each inequation once. We will refer to the algorithm formed by augmenting KBI with this rule as KBC, standing for Knuth-Bendix with *contradiction*, since this new rule involves a proof by contradiction step. We will ultimately describe this in terms of inference rules, avoiding the somewhat recursive use of $\Rightarrow$, but first we must see that the rule as it stands is flawed and how it should be improved.

Our main interest is in the uniqueness of the result of this process. Firstly, for a given system $F$ we would like KBC($F$) to be unique in the sense that the result is the same for each application of KBC, independent of the order of the elements in $F$. Secondly, and more strongly, whenever we have equivalent systems $F_1$ and $F_2$ we would like to have KBC($F_1$) = KBC($F_2$), as we have for equational systems (see Corollary 3.3). This second condition, which obviously implies the first, is essential for proof by invariance and will be important later in our work.

Unfortunately, for KBC as it stands we have neither form of uniqueness, as seen in the following example.

EXAMPLE 3.20 (Multiple minimal systems). Consider the system

$$F = \left\{ \begin{array}{l} \mathsf{a\ b = b},\ \mathsf{a\ a = d},\ \mathsf{a\ d = a}, \\ \mathsf{b\ c \neq a},\ \mathsf{b\ c \neq d} \end{array} \right\}.$$

Both of

$$F_1 = \{\mathsf{a\ b = b},\ \mathsf{a\ a = d},\ \mathsf{a\ d = a},\ \mathsf{b\ c \neq a}\}$$
$$F_2 = \{\mathsf{a\ b = b},\ \mathsf{a\ a = d},\ \mathsf{a\ d = a},\ \mathsf{b\ c \neq d}\}$$

have minimal inequations, with $F_1$ or $F_2$ generated by KBC depending on the ordering of the two inequations in $F$. $\diamond$

The problem is that we cannot ensure the inequations will be processed in the same order. The remedy is to extend our term order to an order on inequations. If $s_1 \neq t_1$ and $s_2 \neq t_2$ are *oriented* inequations, so that $s_1 > t_1$ and $s_2 > t_2$, then $(s_1 \neq t_1) > (s_2 \neq t_2)$ if and only if $s_1 > s_2$, or $s_1 \equiv s_2$

and $t_1 > t_2$. This is in fact our standard ordering where $s_1 \neq s_2$ is written in prefix notation as $\neq \langle s_1, t_1 \rangle$ and $\neq$ is treated as a commutative operator.

With this ordering the above rule for KBC can be refined to

*ReduceU* for each inequation $f \in F$,
from highest to lowest,
if $F \setminus f \Rightarrow f$ then remove $f$ from $F$.

This procedure then satisfies our first uniqueness conditions since for a fixed $F$ the inequations will be processed in a fixed order. However, we still cannot satisfy the second requirement. In Example 3.20 $F_1$ and $F_2$ are equivalent, both being minimal subsets of $F$, but neither can be reduced by KBC and so $\text{KBC}(F_1) \neq \text{KBC}(F_2)$. We must add to this reduction strategy a process of expansion, including inequations $f$ such that $F \Rightarrow f$ in a similar way that *Deduce* is used for equational completion. There is no need to add inequations that are ordered higher than those in $F$ since they would then be removed by *ReduceU*. Instead we want to add to $F$ any inequation $f$ such that $F \Rightarrow f$ and $f$ is less than the highest inequation in $F$. We call this rule *DeduceU*, defined by

*DeduceU* for all inequations $f \notin F$,
such that $f < g$ for some $g \in F$,
if $F \Rightarrow f$ then add $f$ to $F$.

As with reduction, since $\Rightarrow$ is a consequence relation we know for any system $F$ and $f$ such that $F \Rightarrow f$ we have that $F$ and $F \cup f$ are equivalent.

Our final KBC algorithm is then KBI together with the new rules *ReduceU* and *DeduceU*, all of which are repeatedly applied until no further changes are possible. It is clear that if the equational component of a system $F$ is bounded then KBC will terminate, since *ReduceU* only removes inequations and for a given inequation $g$ there are only finitely many $f$ such that $f < g$. More important is the uniqueness of the result, as given by the following theorem.

THEOREM 3.7 (Uniqueness). *If $F_1, F_2 \subseteq \mathcal{F}$ are bounded then $F_1$ and $F_2$ are equivalent if and only if KBC$(F_1)$ = KBC$(F_2)$.*

PROOF. Suppose $F_1$ and $F_2$ are equivalent. Let $F_1^* = \text{KBC}(F_1)$ and $F_2^* = \text{KBC}(F_2)$. Since the inequation processing of KBC does not effect the equation processing, by Corollary 3.3 we have that the rewrite rules of $F_1^*$ and $F_2^*$ are equal. We must now show that the sets of inequations are also equal.

Let $f_1$ be the highest inequation in $F_1^*$, and $f_2$ the highest inequation in $F_2^*$. Suppose that $f_2 > f_1$. Now $F_1^* \Rightarrow f_2$, since $F_1^*$ and $F_2^*$ are equivalent. Applying *DeduceU* to $F_2^*$ involves adding all consequences of $F_2^*$ less than $f_2$. Since $f_1 < f_2$ and $f_1$ is the highest inequation in $F_1^*$, this includes all

inequations of $F_1^*$ (again using the equivalence of $F_1^*$ and $F_2^*$). $f_2$ is still the highest term of this enlarged system and so it is the first examined by *ReduceU*. Since $F_1^* \Rightarrow f_2$ it is then removed, contradicting the minimality of $F_2^*$. If $f_1 > f_2$ we obtain the same contradiction, so that we must have $f_1 = f_2$.

Now consider $f_1'$ and $f_2'$, the second highest inequations in $F_1^*$ and $F_2^*$, and suppose $f_2' > f_1'$. Again $F_1^* \Rightarrow f_2'$ and applying *DeduceU* to $F_2^*$ adds all of $F_1^*$ except $f_1$ (since $f_1 = f_2 > f_2'$) to $F_2^*$. Assuming $F_1^*$ and $F_2^*$, we cannot then remove $f_1$ using *ReduceU*, but $f_2'$ is removed as before. Thus we will have $f_1' = f_2'$.

This process can be continued until we have considered all of the inequations in one of the systems, say $F_1^*$. If the remaining inequations in the other system are given by $G = F_2^* \setminus F_1^*$, then for each $g \in G$ we have $F_1^* \Rightarrow g$, by equivalence, and so $F_2^* \setminus g \Rightarrow g$ since we have established that $F_1^* = F_2^* \setminus G$, contradicting the minimality of $F_2^*$. Thus $F_1^* = F_2^*$.

The converse follows from the soundness of KBC. □

This is our fundamental theorem for inequational reasoning, showing that KBC is a true extension of KB to systems of equations and inequations. It is worth reflecting here on the importance of having a total ordering on terms, the result of our restriction to a variable-free language. This has made possible both this theorem and Theorem 3.1. Here however we are making greater use of the signature's total word ordering by looking at the ordering between terms of different types.

Theorem 3.7 has three important applications in our work. The most immediate of these is to enable us to extend proof by invariance to possible theorems $(F, f)$ where the conclusion $f$ is an inequation. If indeed $F \Rightarrow f$ then $F$ and $F \cup f$ are equivalent and so the result of applying KBC to each will be the same. If the results are not the same then $F$ and $F \cup f$ are not equivalent and so $F \not\Rightarrow f$.

When we apply KBI to $F \cup \neg f$ as part of a proof by contradiction, if we do not obtain $\perp$ then we cannot say anything about the minimality or uniqueness of the resulting output. The second application of this theorem will come in Section 3.5 when we look at finding ways of augmenting the hypotheses of such a non-theorem to give a new theorem. These *side conditions* are obtained from looking at the result of the unsuccessful proof. When we simply want a yes or no answer for a given proof then we will continue to use KBI since it is cheaper, but if we are interested in side conditions then it is preferable to use KBC because we know they will then be unique for a given possible theorem.

The final application of the theorem is to motivate a similar process of finding minimal sets for equational systems. We will loosely call this process *inverse deduction* because it undoes the completion work of the Knuth-Bendix procedure, throwing deductive consequences away rather than adding them. This will be used in Section 6.4 when discussing axioms for theoretical systems.

We conclude this section by specifying the KBC procedure in terms of inference rules, given in Table 3.2. Since a successful proof by contradiction involves only a single inequation we need only mention one, $s_1 \neq t_2$, in the rules *ReduceU* and *DeduceU*. *ReduceU* involves a similar critical pair process to the equational *Deduce*, but *DeduceU* involves a more complicated search for appropriate inequations to add. This is still a simple matter for small signatures, since the number of lesser inequations is then small, but may be harder for larger problems.

The two new rules are placed at the end of the algorithm. Unlike *Contradiction* and *SimplifyU*, they cannot alter the termination of the algorithm by generating an inconsistency. It is also natural to have a canonical system $R$ to use in reducing and deducing the inequations.

| | | | |
|---|---|---|---|
| *Contradiction* | $(E \cup \{s \neq s\}, R)$ | $\Rightarrow$ | $\bot$ |
| *SimplifyU* | $(E \cup \{s \neq t\}, R)$ | $\Rightarrow$ | $(E \cup \{s \neq u\}, R)$ if $t \rightarrow_R u$ |
| *Delete* | $(E \cup \{s = s\}, U, R)$ | $\Rightarrow$ | $(E, U, R)$ |
| *Compose* | $(E, U, R \cup \{s \rightarrow t\})$ | $\Rightarrow$ | $(E, U, R \cup \{s \rightarrow u\})$ if $t \rightarrow_R u$ |
| *Simplify* | $(E \cup \{s = t\}, U, R)$ | $\Rightarrow$ | $(E \cup \{s = u\}, U, R)$ if $t \rightarrow_R u$ |
| *Orient* | $(E \cup \{s = t\}, U, R)$ | $\Rightarrow$ | $(E, U, R \cup \{s \rightarrow t\})$ if $s > t$ |
| *Collapse* | $(E, U, R \cup \{s \rightarrow t\})$ | $\Rightarrow$ | $(E \cup \{u = t\}, U, R)$ if $s \rightarrow_R u$ |
| *Deduce* | $(E, U, R)$ | $\Rightarrow$ | $(E \cup \{s = t\}, U, R)$ if $s \longleftarrow_R u \longrightarrow_R t$ |
| *ReduceU* | $(E, U \cup \{s_1 \neq t_1, s_2 \neq t_2\}, R)$ | $\Rightarrow$ | $(E, U \cup \{s_1 \neq t_1\}, R)$ if $s_2 > t_2$ and $s_1 \longleftarrow_{R \cup \{s_2 \rightarrow t_2\}} u \longrightarrow_{R \cup \{s_2 \rightarrow t_2\}} t_1$ where the inequations $s_2 \neq t_2$ are tried from highest to lowest |
| *DeduceU* | $(E, U \cup \{s_1 \neq t_1\}, R)$ | $\Rightarrow$ | $(E, U \cup \{s_1 \neq t_1, s_2 \neq t_2\}, R)$ if $s_2 > t_2$ and $s_1 \longleftarrow_{R \cup \{s_2 \rightarrow t_2\}} u \longrightarrow_{R \cup \{s_2 \rightarrow t_2\}} t_1$ $(s_2 \neq t_2) < (s_3 \neq t_3)$, for some $(s_3 \neq t_3) \in U \cup \{s_1 \neq t_1\}$ |

TABLE 3.2. Inference rules for KBC

### 3.4.5. Proof by Invariance

EXAMPLE 3.21 (Invariance proof). Consider again Example 3.15, now with proof by invariance using KBC. For illustrative purposes we look at the results from both relative orderings of **male** and **father**. Firstly, if **male** $<$ **father** then KBC applied to $F$ is simply $F$ itself. Applied to $F \cup f$ we find the peak

$$\mathsf{male\ Alice} \longleftarrow_{R'} \mathsf{male\ father\ Alice} \longrightarrow_{R'} \mathsf{True}$$

during *ReduceU*, where $R' = \{\mathsf{male\ father} \to \mathsf{True}\ !,\ \mathsf{father\ John} \to \mathsf{Alice}\}$. Thus $f$ is removed from the system and the theorem is proved.

Alternatively, if **male** $>$ **father** then KBC applied to $F$ uses *DeduceU* to generate the new inequation **father John** $\neq$ **Alice**, from the same peak as above. The original inequation is not removed since we cannot deduce **father John** $=$ **Alice** from

$$R'' = \{\mathsf{male\ father} \to \mathsf{True}\ !,\ \mathsf{male\ Alice} \to \mathsf{True}\}.$$

This system is then $F \cup f$ and so applying KBC to the augmented system gives the same result, again proving the theorem. $\diamond$

Proof by invariance can be implemented quite efficiently, even though *DeduceU* is potentially complicated. Firstly, if we are trying to prove an equation then it is not necessary to examine the inequations at all since they cannot effect the processing of equations. (The one exception is that we should use the inequations to test for inconsistency).

We will see later that in a proof by contradiction it is useful to order all terms appearing in the conclusion higher than those appearing only in the hypotheses. If we follow the same rule here then when we have a theorem whose conclusion is an inequation, proof by invariance using KBC will first remove the conclusion, since it is the highest ordered inequation. Then the states of KBC($F$) and KBC($F \cup f$) will be equal at some stage, and so their final results will be equal by uniqueness, proving the theorem.

### 3.4.6. Inductive Proofs of Inequations

In Definition 3.12 we defined the notion of an inductive theorem, where the inductive conclusion was an equation. For completeness we might also define inductive proof for function inequations. However, whereas proving an equation is an inductive consequence of a system involves proving it must hold for all possible applications of it, to prove an inequation we need only show that indeed the left and right sides are not equal for a single application.

Specifically, to show that $f \neq g$ is an inductive consequence of a system $F$ we need only find one $\Sigma$-entity $a \in \mathrm{dom}(f)$ such that there are two distinct

$\Sigma$-entities $b_1, b_2 \in \text{cod}(f)$ with $fa =_F b_1$ and $ga =_F b_2$. But if this is the case then $\text{KBI}(F \cup (f = g)) = \bot$ and so $f \neq g$ is in fact a *deductive* consequence of $F$ (using proof by contradiction). In accordance with Definition 3.12 we would thus say that there are *no* inductive theorems with an inequation as the conclusion.

## 3.5. Theorems and Possible Theorems

When we present a possible theorem to KBI for proof by contradiction we have two possibilities (assuming the procedure terminates). If the theorem is true then KBI will fail by meeting a contradiction. In this case there is nothing further to be done, and we return the result 'True'.

The result of a successful completion of KBI (where a contradiction is not encountered) will be a set $S^*$ of inequations and rewrite rules (the latter of which can be viewed as equations). In this case the possible theorem, as given, is a non-theorem. However, we can interpret the $S^*$ to gain information as to why it is not true, and from this extend the hypotheses to give a theorem.

### 3.5.1. Side Conditions

Our development here parallels some of that for geometry theorems expressed as rational polynomials, as described in [5].

Note that if $(F \cup \neg f) \Rightarrow g$ then $(F \cup \neg f) \cup \neg g$ reduces to a contradiction. Then trivially so does $(F \cup \neg g) \cup \neg f$ so that $(F \cup \neg g) \Rightarrow f$. Hence any consequence of $F \cup \neg f$ can be added to the hypotheses of the pair $(F, f)$ to produce a theorem. We call such a consequence $g \in \mathcal{F}$ a *side condition* for $(F, f)$, and write $F : f$ for the set of all side conditions for $(F, f)$. Thus we may write

$$F : f = \{g \mid F \cup \neg g \Rightarrow f\}.$$

Note that for the possible theorem $(F, f)$, any $h \in \mathcal{F}$ such that $F \Rightarrow h$ will always be a side condition for $(F, f)$. We call any such $h$, where $h$ is a consequence of the hypotheses alone, an *extraneous* side condition for $(F, f)$. Such conditions are of little use and we will be interested in detecting (and discarding) extraneous conditions in our system.

We call a set of side conditions $G$ for $(F, f)$ *dense* if the deductive consequences of $G$ are exactly $F : f$. That is, $G$ is dense if

$$\{g \in \mathcal{F} \mid G \Rightarrow g\} = \{h \in \mathcal{F} \mid (F \cup \neg f) \Rightarrow h\}.$$

It is obvious from this that $\text{KBI}(F \cup \neg f)$ is a dense set of side conditions.

The following two lemmas give general information about proving possible theorems when the hypotheses consist of equations alone.

LEMMA 3.5. *For $F \subseteq \mathcal{E}$ and $f \in \mathcal{E}$, if $g \in \mathcal{E}$ is a side condition for $(F, f)$ then $g$ is extraneous.*

PROOF. The input to KBI is $F \cup \neg f$, where $\neg f$ is the only inequation. It is clear from the inference rules for KBI that the equations $F$ are processed independently of $\neg f$ and hence any equation that is a consequence of $F \cup \neg f$ must be therefore be a consequence of $F$. □

From this lemma, when the hypotheses and conclusion involve only equations then the only non-extraneous side condition will be the negation of the conclusion, reduced by the canonical system for $F$ generated during KBI. Consider the following:

EXAMPLE 3.22 (Ordering).

$$F = \{\mathsf{father\ sister} = \mathsf{father},\ \mathsf{sister\ Alice} = \mathsf{Lucia}\},$$
$$f = (\mathsf{father\ Alice} = \mathsf{John}).$$

The processing of $F$ results in the critical pair deduction

$$\mathsf{father\ Alice} = \mathsf{father\ Lucia}.$$

If our ordering has Alice > Lucia then we have the useful side condition father Lucia = John. However, if Alice < Lucia then the (unreduced) conclusion is the side condition, a less useful situation. We have from this lemma the general rule that all words appearing in the conclusion should be ordered above those appearing only in the hypotheses, allowing the conclusion to be reduced if possible. ◇

LEMMA 3.6. *For $F \subseteq \mathcal{E}$ and $f \in \neg\mathcal{E}$, then $F \not\Rightarrow f$.*

PROOF. Here the input to KBI consists only of equations, and hence there is no way in which the inference rule *Contradiction* can produce $\perp$. □

In this case we *always* have a non-theorem. Unfortunately, there is nothing that can be said about which of the resulting side conditions are extraneous. When the conclusion is an inequation then adding its negation (an equation) to the hypotheses can result in more complicated interactions through critical pair deduction.

EXAMPLE 3.23 (Non-extraneous condition).

$$F = \{\mathsf{husband\ wife} = \mathsf{i}\},$$
$$f = (\mathsf{wife\ John} \neq \mathsf{Jill}).$$

The result of applying KBI is

$$\{\mathsf{husband\ wife} \to \mathsf{i},\ \mathsf{wife\ John} \to \mathsf{Jill},\ \mathsf{husband\ Jill} \to \mathsf{John}\}.$$

The last of these three is a consequence of the hypothesis and the negated conclusion together, and hence is not extraneous, but in general from outside KBI it is hard to establish whether a new equation is extraneous or not. ◇

EXAMPLE 3.24 (Unbounded non-theorem). Even if the hypotheses of a theorem are unbounded, proof by contradiction will still terminate. However, if we have a non-theorem it is possible that the hypotheses combined with the negated conclusion will be unbounded and the process must be artificially halted. Although in this case the full set of side conditions is infinite, we can still obtain some information by looking at the accumulated rewrite system when the proof was aborted.

Consider the possible theorem with

$$F = \left\{ \begin{array}{l} \mathsf{father\ Jill} = \mathsf{George}, \\ \mathsf{fatherinlaw\ father} = \mathsf{father\ mother} \end{array} \right\},$$
$$f = (\mathsf{father\ father\ mother\ Jill} = \mathsf{father\ Alice}).$$

As noted in Example 3.5, the canonical system for $F$ consists of rules

$$\mathsf{father\ mother}^k\ \mathsf{Jill} \to \mathsf{fatherinlaw}^k\ \mathsf{George}$$

for each $k \geq 1$, none of which reduce $\neg f$ to a contradiction. However, $\neg f$ is at least partly reduced by these rules, and so on aborting KBI we obtain the single non-extraneous side condition

$$\mathsf{father\ fatherinlaw\ George} = \mathsf{father\ Alice}.$$

◇

EXAMPLE 3.25 (Impossible theorem). As a final example, consider the following possible theorem.

$$F = \{\mathsf{male\ Mary} \neq \mathsf{True},\ \mathsf{mother\ Paul} = \mathsf{Mary}\},$$
$$f = (\mathsf{male\ mother\ Paul} = \mathsf{True}).$$

The result of applying KBI is

$$\{\mathsf{male\ Mary} \neq \mathsf{True},\ \mathsf{mother\ Paul} \to \mathsf{Mary}\},$$

the negated conclusion being reduced to $\mathsf{male\ Mary} \neq \mathsf{True}$ and then removed by *Delete*. Thus we have no non-extraneous side conditions. This can only happen when the negated conclusion itself is a consequence of the hypotheses and so is removed during KBI. In this case there is no equation that we can add to the hypotheses to produce a theorem, except one that contradicts the hypotheses and gives a trivial theorem. We thus call such a pathological pair $(F, f)$ an *impossible theorem*. ◇

Beyond the specific cases given above, we must use the original definition to identify which side conditions are extraneous. We find the reduced set $G = \mathrm{KBI}(F \cup \neg f)$ and then for each $g \in G$ determine if the pair $(F, g)$ constitutes a theorem. This can be done quite efficiently for all the equations in $G$ since we need only find a canonical system for the equations of $F$ and

then carry out a normalization proof for each equation $g$. A little more work is needed for the inequations in $G$ as these can result in new critical pair deductions when their negations are added to $F$. However, this can also be kept to a minimum by adding them to KBI($F$) instead.

When implementing KBI or KBC it is also possible to keep track of which rules and inequations are used to generate or reduce others. Those that are the result of hypotheses alone are then necessarily extraneous.

### 3.5.2. Side conditions for Compound Sentences

Proving a possible theorem $(F, f_1 \wedge \cdots \wedge f_n)$ involves $n$ applications of KBI, resulting in $n$ canonical sets $G_1 = \text{KBI}(F \cup \neg f_1), \ldots, G_n = \text{KBI}(F \cup \neg f_n)$. If each $G_j$ is $\perp$ then $(F, f_1 \wedge \cdots \wedge f_n)$ is a theorem. Otherwise its side conditions will be the conjunction of the side conditions from each $G_j \neq \perp$, each of which in turn is a disjunction.

EXAMPLE 3.26 (Compound side conditions). Consider the non-theorem of Example 3.18 where two subproofs were required and one did not succeed. Although the conclusion is compound, since one part did succeed we only need a side condition for the remaining subtheorem. The corresponding $G$ is

$$G = \left\{ \begin{array}{l} \mathsf{father\ sister} \rightarrow \mathsf{father},\ \mathsf{father\ Lucia} \rightarrow \mathsf{John}, \\ \mathsf{father\ Alice} \rightarrow \mathsf{John},\ \mathsf{sister\ Alice} \neq \mathsf{Lucia} \end{array} \right\}.$$

The last two elements of $G$ are the unprocessed conclusion of the subtheorem and so the original theorem has the non-extraneous side conditions $\mathsf{father\ Alice} \neq \mathsf{John}$ and $\mathsf{sister\ Alice} = \mathsf{Lucia}$. We can also write this as the single compound condition

$$(\mathsf{father\ Alice} \neq \mathsf{John}) \vee (\mathsf{sister\ Alice} = \mathsf{Lucia}).$$

◇

EXAMPLE 3.27 (Compound side conditions). We can extend Example 3.23 to a possible theorem with

$$F = \{(\mathsf{husband\ wife} = \mathsf{i}) \vee (\mathsf{husband\ Jill} = \mathsf{John})\},$$
$$f = (\mathsf{husband\ Jill} = \mathsf{John}) \wedge (\mathsf{wife\ John} \neq \mathsf{Jill}).$$

Proving this requires four subproofs. For each of these Table 3.3 shows if it succeeds or otherwise gives the corresponding non-extraneous side conditions.

Thus the possible theorem $(F, f)$ has the compound side condition

$$g_1 \wedge (\neg g_1 \vee \neg g_2) \wedge \neg g_2,$$

| Hypotheses | Conclusion | Result |
|---|---|---|
| {husband wife = i} | husband Jill = John | husband Jill = John |
| {husband wife = i} | wife John ≠ Jill | (wife John ≠ Jill) ∨ (husband Jill ≠ John) |
| {husband Jill = John} | husband Jill = John | True |
| {husband Jill = John} | wife John ≠ Jill | wife John ≠ Jill |

TABLE 3.3. Results of subproofs for Example 3.27

where $g_1$ = (husband Jill = John) and $g_2$ = (wife John = Jill). This can be readily expanded to

$$\begin{aligned}((g_1 \wedge \neg g_1) \vee (g_1 \wedge \neg g_2)) \wedge \neg g_2 &= g_1 \wedge \neg g_2 \wedge \neg g_2 \\ &= g_1 \wedge \neg g_2.\end{aligned}$$

This give the simplified side condition

$$\text{(husband Jill = John)} \wedge \text{(wife John} \neq \text{Jill)}.$$

Thus to prove the conclusion we need to know the conclusion. Even though we could prove one solution, the hypotheses in essence contain no information about the conclusion. This is the 'I don't know' response of Example 4.3. ◇

EXAMPLE 3.28 (Impossible conjunction). Consider a possible theorem similar to that in Example 3.25.

$$F = \left\{ \begin{array}{l} \text{male Mary} \neq \text{True, wife Peter = Mary,} \\ \text{husband wife = i, male husband = True !} \end{array} \right\},$$
$$f = \text{(male Peter = True)} \wedge \text{(male wife Peter = True)}.$$

The first part of the conjunction is proved but the second has no non-extraneous side conditions and so is impossible. If we can never prove one part of the conjunction then we can never prove the conjunction itself so that the whole theorem is impossible. ◇

EXAMPLE 3.29 (Hypotheses with implication (revisited)). As a final example, recall the non-theorem of Example 3.19, with

$$F = \{\text{(father Alice = John)} \rightarrow \text{(male John = True)}\},$$
$$f = \text{(male John = True)},$$

where we were unable to prove the consequent of an implication. The subproof with hypotheses {male John = True} was trivially a theorem, but that with {father Alice ≠ John} gave no reduction with KBI. We thus obtain the single side condition father Alice = John. Even though this is a consequence of the hypotheses of its subtheorem, it is *not* a consequence of the original $F$ and hence is non-extraneous.

In general, the possible theorem $(\{f \rightarrow g\}, g)$ will have the non-extraneous side condition $f$. In these terms we can view $f$ as an *abductive explanation* of an observation $g$ based on the knowledge that $f \rightarrow g$. We will expand on this idea in the following section. $\diamond$

## 3.6. Abductive Reasoning

We view a reasoner as having a set of beliefs $F$ from which it seeks to prove a statement $f$. If it fails to find a constructive proof of $f$ from $F$ then it does not have any belief in $f$. The side conditions of the non-theorem $(F, f)$ then give conditions under which a theorem would exist. We use these in the next chapter with a dialogue between an oracle and the reasoner. There the statement $f$ is a question from the oracle; if the reasoner cannot answer that $f$ is indeed a consequence of its knowledge (or an impossibility), then it responds with the conditions that would make $f$ hold.

In this manner we view side conditions as experiments needed to answer a particular questions. In line with the discussion below, we call such conditions *abductive experiments*. We will also look at experiments in other contexts (see Sections 4.2.4 and 6.2.3).

Side conditions can also be viewed as *abductive hypotheses* [**37**]. The abductive inference rule says that if we know $x \Rightarrow y$ and we know $y$, then we can infer $x$. The process of abductive reasoning then takes belief in $x \Rightarrow y$ and belief in $y$ to infer tentative belief in the abductive hypothesis $x$. In terms of our equational proof, if we believe $F$ and $f$, and can obtain the theorem $(F \cup G) \Rightarrow f$, then we hypothesize belief in the side conditions $G$. This is quite different from the view of theorem proving where we have no tendency to believe the generated conditions.

EXAMPLE 3.30 (Diagnosis). A standard example of abductive reasoning is medical diagnosis [**10**]. The system $F$ gives medical knowledge and perhaps known information about a patient, with $f$ giving observed symptoms. The abductive side conditions then give possible explanations of the symptoms.

The types of side conditions we have identified have useful interpretations here. An extraneous side condition corresponds to an explanation which does not involve the symptoms themselves and so is of little use to the doctor. If the only non-extraneous side conditions are simply the symptoms then there is no explanation for $f$ present in $F$.

Finally, an impossible theorem indicates that on the basis of $F$, the patient should *not* have the symptoms $f$, suggesting the symptoms be rechecked or that $F$ be somehow revised. At the opposite extreme, we may indeed find $F \Rightarrow f$, so that the symptoms are *expected* (logically necessary) from the information in $F$.

In all, proof by contradiction provides an immediate logical system for examining abductive reasoning.

Consider the signature

$$\Sigma = \left\{ \begin{array}{l} \mathsf{patient} \in \mathsf{Person}, \\ \mathsf{Flu}, \mathsf{Mumps} \in \mathsf{Illness}, \\ \mathsf{True}, \mathsf{False} \in \mathsf{Sentence}, \\ \mathsf{illness} : \mathsf{Person} \to \mathsf{Illness}, \\ \mathsf{sneezing} : \mathsf{Person} \to \mathsf{Sentence} \end{array} \right\},$$

and the medical knowledge

$$F = \{(\mathsf{illness\ patient} = \mathsf{Flu}) \to (\mathsf{sneezing\ patient} = \mathsf{True})\}.$$

Making the observation $f = (\mathsf{sneezing\ patient} = \mathsf{True})$, we attempt a proof of $F \Rightarrow f$. This is equivalent to Example 3.29, resulting in the single non-extraneous side condition $\mathsf{illness\ patient} = \mathsf{Flu}$. Here we interpret this condition, that the patient has flu, as an abductive explanation of the symptom, the sneezing patient. $\diamond$

Cox and Pietrzykowski [**10**] give four conditions that a 'useful' abductive hypothesis should satisfy. These are indeed satisfied by our use of side conditions. Two of the requirements, that a hypothesis be *minimal* and *basic*, are immediate consequences of using the canonical form algorithm of KBC.

An abductive hypothesis which is *consistent* with the knowledge corresponds precisely to our notion of a non-extraneous side condition. Selecting only *non-trivial* hypotheses, those which do not immediately imply the symptoms, matches our observations in Examples 3.27 and 4.3.

Thus the procedure for generating side conditions gives an immediate system for abductive reasoning. However, we will continue to use side conditions from our original deductive viewpoint. Applications of the abductive belief method will be left to future work.

## 3.7. Solving equational systems

The idea behind side conditions, of finding information needed to give a theorem, is similar to that of solving numerical equations. In the latter, for example, we might have a system of equations $G = \{g_1(x) = b_1, \ldots, g_n(x) = b_n\}$ in some variable $x$. If we can find an equation $x = a$ such that $G \Rightarrow (x = a)$ then we would say that $x = a$ is a *solution* of $G$.

We can make a similar formulation for our term-based reasoning. Let $F = \{f_1, \ldots, f_n\}$ be a system of equations and inequations involving an indeterminate $\mathsf{x}$. Then if $F \Rightarrow (\mathsf{x} = a)$ for some $a$ then we say that $\mathsf{x} = a$ is a *solution* of the system $F$.

Suppose then that $g$ is a solution of some system $F \cup f$, where $f$ involves an indeterminate. Then $F \cup f \Rightarrow g$ and so $\mathrm{KBI}((F \cup f) \cup \neg g) = \bot$. Rearranging gives $\mathrm{KBI}((F \cup \neg g) \cup f) = \bot$ so that $F \cup \neg g \Rightarrow \neg f$. Thus the negation of $g$ will be a side condition of the possible theorem $(F, \neg f)$, so we can find a solution of $F \cup f$ by trying to prove $F \Rightarrow \neg f$. If there are no indeterminates in $F$ then no extraneous side condition can involve an indeterminate. The non-extraneous side conditions can then be used to give information about the solution.

This is a somewhat artificial use of side conditions, and is limited by the number of equations that can involve indeterminates and the number of distinct indeterminates present in the system. But it shows that to solve a system of equations $F$ we need only to find a canonical system for $F$ using KBI. We can then look at the elements of $\mathrm{KBI}(F)$ which involve indeterminates and obtain information from them about the solution of $F$.

EXAMPLE 3.31 (Solving). Consider the system

$$F = \left\{ \begin{array}{l} \mathsf{father\ sister} = \mathsf{father},\ \mathsf{sister\ Alice} = \mathsf{Lucia}, \\ \mathsf{father\ Lucia} = \mathsf{John},\ \mathsf{father\ Alice} = \mathsf{x} \end{array} \right\}.$$

We want to find a solution for $F$ in terms of the indeterminate $\mathsf{x} \in \mathsf{person}$. Applying KBI to $F$ gives only one equation, $\mathsf{x} = \mathsf{John}$, involving an indeterminate, the solution of the system. $\diamond$

Note that the equation $\mathsf{father\ Alice} = \mathsf{x}$ is a very simple equation to solve. We are merely finding the reduced form of $\mathsf{father\ Alice}$ under the given hypotheses. A much more difficult equation to solve would be $\mathsf{father\ x} = \mathsf{John}$. The left-hand side $\mathsf{father\ x}$ is not reduced by the above $F$ and so no useful side conditions exist, giving no solution to the equation. The reason for this is two-fold, as outlined in the following examples.

EXAMPLE 3.32 (Inverse functions). Firstly consider the simpler system

$$F = \{\mathsf{husband\ Mary} = \mathsf{Peter},\ \mathsf{husband\ x} = \mathsf{Peter}\}.$$

This time the only element in the canonical system involving an indeterminate is $\mathsf{husband\ x} = \mathsf{Peter}$. We have been unable to make any progress in solving the equations. This is a nice illustration of the differences between our functional approach and a method based on unification where we could have unified $\mathsf{x}$ with $\mathsf{Mary}$.

What we are missing here is obvious from the mathematical problem of finding an $x$ such that $f(x) = b$. There if $f$ has an inverse $f^{-1}$ then we have $f^{-1}(f(x)) = f^{-1}(b)$ so $x = f^{-1}(b)$. For our problem we can extend the system $F$ to similarly include information about the inverse of $\mathsf{husband}$, giving the new

$$F' = F \cup \{\mathsf{wife\ husband} = \mathsf{i}\}.$$

Completion of $F'$ generates wife Peter $\rightarrow$ Mary and wife Peter $\rightarrow$ x, which then collapses to give x $\rightarrow$ Mary. This new rule eliminates the original indeterminate equation, giving the solution x $\rightarrow$ Mary.

Note that, as with the motivating mathematical case, we only needed a left inverse of husband. $\diamond$

EXAMPLE 3.33 (Multi-valued functions). Turning to the harder problem

$$F = \left\{ \begin{array}{l} \text{father sister} = \text{father}, \ \text{sister Alice} = \text{Lucia}, \\ \text{father Lucia} = \text{John}, \ \text{father x} = \text{John} \end{array} \right\},$$

we again require information about the inverse to find a solution. The natural inverse of father : person $\rightarrow$ person is the multivalued function child : person $\rightarrow$ {person}. But the type of the child father is then person $\rightarrow$ {person} so we cannot form an equation child father = i. The problem is that if the inverse is to be multivalued then the original function must have a multi-element domain, and so we instead look at the function father : {person} $\rightarrow$ person. Consider firstly the modified

$$F' = \left\{ \begin{array}{l} \text{father } \{\text{Alice}\} = \text{John}, \ \text{father } \{\text{Lucia}\} = \text{John}, \\ \text{child father} = \text{i}, \ \text{father x} = \text{John} \end{array} \right\}.$$

In completing $F'$ we deduce child John $\rightarrow$ {Alice} and child John $\rightarrow$ {Lucia}, giving the contradictory {Alice} = {Lucia}, indicating that $F'$ is inconsistent.

This illustrates the burden imposed by confluence on the use of sets, forcing all entity values of a multivalued function to be explicitly defined in a single equation. Here we have defined values of child by giving two rules for father, causing inconsistency when the inverse is introduced. To make $F'$ consistent we must rephrase it as

$$F'' = \left\{ \begin{array}{l} \text{father } \{\text{Alice}, \text{Lucia}\} = \text{John}, \\ \text{child father} = \text{i}, \ \text{father x} = \text{John} \end{array} \right\},$$

giving the solution x $\rightarrow$ {Alice, Lucia}. The process is now identical in structure to Example 3.32. $\diamond$

So far we have looked at a system with only a single equation involving an indeterminate. The following three examples extend the simple system $F$ given in Example 3.31 to illustrate the possibilities when we have a system of many equations in a single unknown.

EXAMPLE 3.34 (Inconsistent system). We have already discussed the inconsistency of a system in general and the idea is identical here. If we solve a system by applying KBI to it and we obtain $\perp$ then we say the system is *inconsistent*. For example, if

$$F_1 = F \cup \left\{ \begin{array}{l} \text{wife father} = \text{mother}, \ \text{wife John} = \text{Jill}, \\ \text{mother Alice} = \text{x} \end{array} \right\}$$

then reduction produces the two rules $\mathsf{x} \to \mathsf{John}$ and $\mathsf{x} \to \mathsf{Jill}$, giving the contradiction $\mathsf{John} = \mathsf{Jill}$. We need not always get a contradiction in this way, as seen in Example 3.35.

Note that inconsistency can arise here independent of the equations involving indeterminates, perhaps a little different to the standard notion of an inconsistent system of equations in mathematics. For example, the system

$$\left\{ \begin{array}{l} \mathsf{father\ Alice} = \mathsf{John},\ \mathsf{father\ Alice} = \mathsf{Peter}, \\ \mathsf{mother\ Alice} = \mathsf{x} \end{array} \right\}$$

is inconsistent without the indeterminate in $\mathsf{mother\ Alice} = \mathsf{x}$. We will call a system involving indeterminates *trivially inconsistent* if indeed it remains inconsistent when all equations involving indeterminates are removed. ◇

EXAMPLE 3.35 (Redundant equations). Consider the extension

$$F_3 = F \cup \{\mathsf{father\ Lucia} = \mathsf{x}\}.$$

Here we obtain through either equation involving x the new rule $\mathsf{x} \to \mathsf{John}$. This then reduces the two equations to $\mathsf{father\ Alice} = \mathsf{John}$ and $\mathsf{father\ Lucia} = \mathsf{John}$. The latter of these is already present in $F_3$ and so is deleted. Thus the canonical form of $F_3$ is the same as the canonical form of $F$ and so the additional equation is *redundant*. ◇

EXAMPLE 3.36 (Eliminating unknowns). Finally, consider

$$F_4 = F \cup \{\mathsf{wife\ x} = \mathsf{Jill}\}.$$

This is similar to the previous example in that we get the consistent solution $\mathsf{x} \to \mathsf{John}$. However the additional equation this time is not redundant, containing information not present in $F$. The result instead is a simple elimination of unknowns. ◇

These examples extend to system involving more than one indeterminate. For each indeterminate x in a system $F$ we define a solution for x as before. We then say $F$ has *no* solution if none of the indeterminates in $F$ have a solution. The following result gives an immediate test for the solvability of system.

PROPOSITION 3.3. *If all equations in a system $F$ involving an indeterminate have an indeterminate on both sides then $F$ has no solution.*

PROOF. To obtain a solution the canonical form of $F$ must contain an equation with an indeterminate on only one side. If all equations have indeterminates on both sides then clearly the inference rules *Compose*, *Simplify*, and *Collapse* cannot achieve this alone, since each simply replaces left with right. If the peak $u$ considered by *Deduce* has no indeterminates then it similarly cannot produce a critical pair involving indeterminates on either side. Again if $u$ does involve indeterminates then these cannot be removed. Thus $F$ cannot be solved. □

EXAMPLE 3.37 (Two indeterminates). Consider the system

$$F = \left\{ \begin{array}{l} \mathsf{father\ x = y,\ husband\ Jill = y,} \\ \mathsf{wife\ John = Jill,\ mother\ Alice = Jill,} \\ \mathsf{husband\ mother = father,\ husband\ wife = i,} \\ \mathsf{child\ father = i} \end{array} \right\}.$$

Finding the canonical form of $F$ gives $\mathsf{x = Alice}$ and $\mathsf{y = John}$. The above theorem implies that solving a system of equations essentially involves finding a solution for one variable and then substituting it to find a solution for the next variable. $\diamond$

We mentioned in passing that with variables and the process of unification we would not need the inclusion of inverses to solve equational systems. Indeed the two main uses of a variable is mathematics give a further illustration of the differences between our rewrite-based reasoning and logic programming. In an equation $2x + y = 4$ the variables $x$ and $y$ take the role of unknowns. We seek values for these unknowns, *bindings* of the variables, which are a model for the equation. On the other hand, in an equation $f(x) = x^2$, the variable $x$ is merely a dummy used to define the function $f$. It is usual to abstract from this definition the function $f$ as an object in itself and reason about it independently of its ground representation. This is the motivation for our variable-free reasoning.

Consider the function equation

$$f = (\mathsf{wife\ father = mother})$$

and the two data $\mathsf{father\ Alice = John}$ and $\mathsf{wife\ John = Jill}$. From these three equations we can then deduce the equation $\mathsf{mother\ Alice = Jill}$. This is a syntactic deduction since in terms of the underlying category the equations are simply identifying morphisms. On the other hand, suppose we have the logic program

$$P = \left| \begin{array}{l} \mathsf{wife}(X, Y) \leftarrow \mathsf{father}(Z, X), \mathsf{mother}(Z, Y) \\ \mathsf{father(alice, john)} \\ \mathsf{wife(john, jill)} \end{array} \right.$$

For any model of $P$, the last two clauses must be true and hence by the first clause we must also have $\mathsf{mother(alice, jill)}$ being true, proving the same result as before.

Yet the equation $f$ is stronger than the first clause of $P$, expressing an equality of functions rather than a unidirectional implication of predicates. If we instead had the two data $\mathsf{father\ Alice = John}$ and $\mathsf{mother\ Alice = Jill}$ we could then prove $\mathsf{wife\ John = Jill}$, while the corresponding change in the unit clauses of $P$ would not be able to prove this. We will see in the applications of Section 5.3 that this extra strength has a significant effect on our reasoning abilities.

The final combination of data, with wife John = Jill and mother Alice = Jill, shows the weakness of the functional approach. Together with $f$ we are unable to prove father Alice = John, whereas the logic program

$$P' = \left| \begin{array}{l} \mathsf{father}(X, Y) \leftarrow \mathsf{wife}(Y, Z), \mathsf{mother}(X, Z) \\ \mathsf{wife}(\mathsf{john}, \mathsf{jill}) \\ \mathsf{mother}(\mathsf{alice}, \mathsf{jill}) \end{array} \right.$$

makes the binding $X$ = alice and $Y$ = john. For term reasoning we are asking for the person whose father is John and, as we saw in Section 3.7, this requires knowledge of inverse functions, just as in mathematics.

CHAPTER 4

# Logic and Belief

In this chapter we formalize the notions of proof described previously, using standard logical constructions [44]. We again focus on proof by contradiction, defining the proof operation $\Rightarrow$ by

$$F \Rightarrow f \text{ if and only if } \mathrm{KBI}(F \cup \neg f) = \bot.$$

We show that $\Rightarrow$ is in fact a deductive consequence relation over the propositional language $L_\Sigma$ (to be defined below). This is then used to outline a *belief* framework, giving both a static and dynamic view of equational knowledge.

In talking about beliefs we begin to describe an implementation of the proof techniques discussed so far. The two paradigms of equational truth to be presented give rise naturally to different implementations of a rewrite-based reasoning system.

## 4.1. External Truth: Logic of Equations

Given a set of terms $T_\Sigma$ generated by some signature $\Sigma$, we define the set of *atomic sentences* $P_\Sigma$ to be

$$P_\Sigma = \{t_1 = t_2 \mid t_1, t_2 \in T_\Sigma\}.$$

The *propositional language* for $\Sigma$, denoted $L_\Sigma$, is the pair $\langle P_\Sigma, O \rangle$, where the $P_\Sigma$ are the atomic sentences and $O$ is the set of operators $\{\neg, \wedge, \vee, \rightarrow, \equiv, \oplus, \bot\}$. The *sentences* of $L_\Sigma$, which we will also denote by $L_\Sigma$, are defined from the atomic sentences as follows:

- If $A$ is a sentence then so is $\neg A$.
- If $A$ and $B$ are sentences then so are $A \wedge B$, $A \vee B$, $A \rightarrow B$, $A \equiv B$, and $A \oplus B$.

The nullary operator $\bot$ is necessarily a sentence. The sentence $\neg(t_1 = t_2)$ is the inequation we have written as $t_1 \neq t_2$.

### 4.1.1. Quantification

Consider the function equation $f = g$, with $f, g : \sigma \rightarrow \tau$ and $f > g$. This equation is oriented to give the rewrite rule $f \rightarrow g$, and so for any $x \in \sigma$,

$fx \to gx$, giving the same normal form for both $fx$ and $gx$, so that $fx = gx$. Hence the function equation $f = g$ embodies the *universal* predicate

$$\forall x.fx = gx.$$

Similarly, for the inequation $f \neq g$ there must be some $x \in \sigma$ such that the normal forms of $fx$ and $gx$ are different. Thus $f \neq g$ corresponds to the *existential* predicate

$$\exists x.fx \neq gx.$$

Using these observations in reverse gives a basis for inductively generating equations and inequations from observed function behaviour. This process will be examined in Chapter 5.

### 4.1.2. Consequence Relations

We represent knowledge by a set $\Gamma \subseteq L_\Sigma$ of sentences which we know. Our initial knowledge $\Gamma_0$ consists of the atomic sentences $s = s$ for all $s \in T_\Sigma$ and the inequations $s \neq t$ for all $\Sigma$-entities $s, t$ where $s \not\equiv t$.

This description of knowledge makes no requirement of *rationality*. The standard notion of rationality comes from a consequence relation. We will then combine this with knowledge to give a description of *belief*.

DEFINITION 4.1. A *consequence relation* over a language $L$ is a relation, written $\vdash$, between $\mathcal{P}(L)$ and $L$ which satisfies the following properties:

1. Inclusion: if $A \in \Gamma$ then $\Gamma \vdash A$.
2. Monotonicity:
$$\frac{\Gamma \vdash A}{\Gamma \cup \Delta \vdash A}.$$
3. Cut:
$$\frac{\Gamma \vdash C \quad \Delta \cup C \vdash A}{\Delta \cup \Gamma \vdash A}.$$

If a proof by contradiction of $(F, f)$ succeeds then we let $R_F$ denote the rewrite system generated by KBI at the time $\perp$ was found.

THEOREM 4.1. *$\Rightarrow$ is a consequence relation over the propositional language $\langle P_\Sigma, \{\neg\}\rangle$.*

PROOF. 1. If $A \in \Gamma$, to prove $\Gamma \Rightarrow A$ we are applying KBI to the set with both $A$ and $\neg A$. One of these will have the form $(l = r)$ so at some stage during KBI we must have an $R$ such that $l \leftrightarrow_R^* r$. Thus KBI will reduce the other, of the form $(l \neq r)$, to a contradiction.

2. If $\Gamma \Rightarrow A$ then KBI reduces $\Gamma \cup \neg A$ to a contradiction. Thus the rewrite system $R_\Gamma$ generated by KBI involves a rewrite proof $l \leftrightarrow^*_{R_\Gamma}$ which reduces an inequation $(l \neq r)$. The added elements of $\Delta$ may create a contradiction with either $\Gamma$ or $\neg A$, but then we trivially still have $\Gamma \cup \Delta \Rightarrow A$. Otherwise, together with $\Delta$ the resulting rewrite system will still give $l \leftrightarrow^*_{R_\Gamma}$ so that again $\Gamma \cup \Delta \Rightarrow A$.
3. Since $\Gamma \Rightarrow C$, if $C$ is of the form $(l = r)$ then $R_\Gamma$ must reduce $(l \neq r)$ to a contradiction, so that $R_\Gamma \Rightarrow (l = r)$, i.e. $R_\Gamma \Rightarrow C$. Thus if $C$ is used in reducing $\Delta \cup C \cup \neg A$ to a contradiction, then $\Gamma \cup \Delta \cup \neg A$ will also reduce to a contradiction. If $C$ is not involved then $\Delta$ alone is responsible, and again we have $\Gamma \cup \Delta \Rightarrow A$.
   If $C$ is of the form $(l \neq r)$ then $R_{\Gamma \cup \neg C}$ reduces something in $\Gamma$, $D$ say, to a contradiction. Since $\Delta \cup C \Rightarrow A$, if $A$ is of the form $(s = t)$ then $\Delta$ must reduce either $C$ or $\neg A$ to a contradiction. If it is $\neg A$ then we immediately have $\Gamma \cup \Delta \Rightarrow A$; otherwise to reduce $C$ we will have $R_\Delta \Longrightarrow \neg C$. But since $R_{\Gamma \cup \neg C}$ reduces $D$ to a contradiction, $R_{\Gamma \cup \Delta}$ will also, giving $\Gamma \cup \Delta \Rightarrow A$.
   Finally, if $A$ is of the form $(s \neq t)$ then $R_{\Delta \cup \neg A}$ reduces something in $\Delta \cup C$ to a contradiction. If it is in $\Delta$ then $\Delta \Rightarrow A$, so by monotonicity $\Gamma \cup \Delta \Rightarrow A$. Otherwise $R_{\Delta \cup \neg A} \Longrightarrow \neg C$ so that $R_{\Gamma \cup \Delta \cup \neg A}$ reduces $D$ to a contradiction, giving $\Gamma \cup \Delta \Rightarrow A$.

□

Thus the original proof by contradiction method, as given in Section 3.4.1, gives a consequence relation over the language of equations and inequations. The extension of this result to general propositions mirrors the corresponding extension of the proof method.

COROLLARY 4.1. *$\Rightarrow$ is a consequence relation over $L_\Sigma$.*

PROOF. We give proofs for single instances of $\vee$ and $\wedge$. Algorithmically, the extension to general sentences involves expressing them in conjunctive and disjunctive normal forms, and so extended these results simply requires the consideration of further subproofs.

1. If $(f \vee g) \in \Gamma$ then to prove $\Gamma \Rightarrow (f \vee g)$ we must prove the two subtheorems $\Gamma \setminus (f \vee g) \cup f \Rightarrow (f \vee g)$ and $\Gamma \setminus (f \vee g) \cup g \Rightarrow (f \vee g)$. For these we apply KBI to $\Gamma \setminus (f \vee g) \cup \{f, \neg f, \neg g\}$ and $\Gamma \setminus (f \vee g) \cup \{g, \neg f, \neg g\}$, both of which give $\bot$.
   Similarly, if $(f \wedge g) \in \Gamma$ then to prove $\Gamma \Rightarrow (f \wedge g)$ we apply KBI to $\Gamma \cup \neg f$ and $\Gamma \cup \neg g$, again obtaining $\bot$ for both.
2. If $\Gamma \Rightarrow (f \vee g)$ then $\mathrm{KBI}(\Gamma \cup \{\neg f, \neg g\}) = \bot$, so we have that $\Gamma \Rightarrow f$, $\Gamma \Rightarrow g$, or $\{\neg f, \neg g\}$ is inconsistent. If this last case is true then trivially for any $\Delta$ we have $\Gamma \cup \Delta \Rightarrow (f \vee g)$. Otherwise for any $\Delta$, by the

monotonicity of Theorem 4.1, we thus have $\Gamma \cup \Delta \Rightarrow f$ or $\Gamma \cup \Delta \Rightarrow g$, and so $\Gamma \cup \Delta \Rightarrow (f \vee g)$.
If $\Gamma \Rightarrow (f \wedge g)$ then $\Gamma \Rightarrow f$ and $\Gamma \Rightarrow g$. Again by Theorem 4.1 we have that $\Gamma \cup \Delta \Rightarrow f$ and $\Gamma \cup \Delta \Rightarrow g$ for any $\Delta$, giving $\Gamma \cup \Delta \Rightarrow (f \wedge g)$.

3. Suppose $\Gamma \Rightarrow (f \vee g)$, so again $\Gamma \Rightarrow f$ or $\Gamma \Rightarrow g$. (We ignore the case where $f$ is $\neg g$ since the result is then trivial). Similarly, if $\Delta \cup (f \vee g) \Rightarrow (h \vee k)$ then $\Delta \cup (f \vee g) \Rightarrow h$ or $\Delta \cup (f \vee g) \Rightarrow k$. If $\Delta \cup (f \vee g) \Rightarrow h$ then $\Delta \cup f \Rightarrow h$ and $\Delta \cup g \Rightarrow h$, so whether $\Gamma \Rightarrow f$ or $\Gamma \Rightarrow g$ we have $\Delta \cup \Gamma \Rightarrow h$, giving $\Delta \cup \Gamma \Rightarrow (h \vee k)$.
Similarly, if $\Delta \cup (f \vee g) \Rightarrow k$ then $\Delta \cup \Gamma \Rightarrow k$, again giving $\Delta \cup \Gamma \Rightarrow (h \vee k)$.
Alternatively, suppose $\Gamma \Rightarrow (f \vee g)$ and now $\Delta \cup (f \vee g) \Rightarrow (h \wedge k)$. This second theorem requires the four subtheorems $\Delta \cup f \Rightarrow h$, $\Delta \cup f \Rightarrow k$, $\Delta \cup g \Rightarrow h$, and $\Delta \cup g \Rightarrow k$. Thus whether $\Gamma \Rightarrow f$ or $\Gamma \Rightarrow g$, we have $\Delta \cup \Gamma \Rightarrow (h \wedge k)$.
The two cases with $\Gamma \Rightarrow (f \wedge g)$ hold similarly.

□

A consequence relation $\vdash$ is said to be *compact* if whenever $\Gamma \vdash A$ there is a finite subset $\Gamma'$ of $\Gamma$ such that $\Gamma' \vdash A$.

LEMMA 4.1. *$\Rightarrow$ is compact.*

PROOF. If $\Gamma \Rightarrow A$ then proof by contradiction of the possible theorem $(\Gamma, A)$ terminates in a finite number of steps, producing $\perp$. We can then take as $\Gamma'$ the sentences of $\Gamma$ which were used by KBI. □

A consequence relation $\vdash$ is *deductive* if there is a binary operator $\rightarrow$ in $L$ such that for all sentences $A$ and $B$ and sets of sentences $\Gamma$,

$$\Gamma \vdash A \rightarrow B \text{ iff } \Gamma \cup A \vdash B.$$

LEMMA 4.2. *$\Rightarrow$ is deductive.*

PROOF. This result is a simple consequence of the proof method for compound sentences. To prove $\Gamma \Rightarrow (A \rightarrow B)$ we reduce the conclusion to conjunctive normal form, giving $\neg A \vee B$, and then apply KBI to $\Gamma \cup \{A, \neg B\}$. But this can be regrouped as $(\Gamma \cup A) \cup \neg B$, so we are equivalently proving $\Gamma \cup A \Rightarrow B$. □

## 4.2. Belief

The standard notion of a *belief set* [**20**] is a pair $\langle \Gamma, \vdash \rangle$, where $\Gamma$ is a set of sentences and $\vdash$ is a consequence relation. The set $\Gamma$ must be *consistent*, so that $\Gamma \nvdash \perp$, and *closed*, so that whenever $\Gamma \vdash f$ then $f \in \Gamma$. Because of this second condition $\Gamma$ will typically be infinite, making belief sets an impractical model for computation. Hansson and others cite a further limitation of the

belief set model of standard AGM. Paraphrasing an example in [24], if we have acquired the knowledge

$$F = \{\mathsf{capital\ France} = \mathsf{Paris},\ \mathsf{contents\ Fridge} = \{\mathsf{Milk}\}\}$$

then we must also believe

$$f = (\mathsf{capital\ France} = \mathsf{Paris}) \equiv (\mathsf{contents\ Fridge} = \{\mathsf{Milk}\}),$$

since $F \Rightarrow f$. As a belief set we would treat these three equations with equal weight. On finding that the fridge no longer contained milk we are forced to revise our belief set, specifically removing $\mathsf{contents\ Fridge} = \{\mathsf{Milk}\}$. However, we cannot also maintain both the remaining equations, since these together imply the removed equation. As a belief set we might then remove $\mathsf{capital\ France} = \mathsf{Paris}$, leaving belief only in the equation of which we have no direct experience. This suggests that for practical purposes we treat *basic* beliefs and *derived* beliefs separately, and only work with basic beliefs when incorporating new information.

Thus we focus our attention on *bases* for belief sets. We say $\langle \Gamma^*, \vdash \rangle$ is a *basis* for $\langle \Gamma, \vdash \rangle$ if $\Gamma^* \nvdash \bot$ and for any $f \in \Gamma$, $\Gamma^* \vdash f$. We can view our equational systems of Chapter 3 as bases for implicit belief sets, where we have used $\Rightarrow$ for $\vdash$ (a consequence relation by Theorem 4.1). Indeed throughout this work we will use belief set to mean belief basis. This is justified because $\Rightarrow$ embodies a semidecision procedure which allows us to determine if a sentence belongs to the belief set or (usually) if it doesn't belong.

In fact we can do much better than the simple set representation of beliefs by using the basis formalism. The generation of side conditions gives a measure of how close a non-belief is to being in the belief set. If the corresponding theorem is impossible then the belief set cannot be extended to include the non-belief. Otherwise the non-extraneous side conditions give a characterization of extensions to the belief set which include the non-belief. These principles are the basis of our use of side conditions in reasoning.

It is worth noting the distinction between the canonical rewrite system for an equational theory, a basis in the mathematical sense, and a belief basis. Specifically, we can represent a belief basis by giving a canonical system for the equations and inequations it contains. This would be a natural step to take for determining membership of the belief set but we should not discard the original beliefs for reasons similar to those in the above example. As outlined above, when it comes to revising beliefs it is important to know exactly which equations are basic and which are derived. In finding the canonical system we generate derived beliefs and possibly remove the basic ones, destroying the needed information.

### 4.2.1. Implementing Belief: Dialogue and Proof

In the case of external truth, our current knowledge is described by a belief set (basis) $\Gamma$. A reasoning system based on such truth will thus have some kind of belief set itself, from which it can make deductions. But how does our reasoner learn this knowledge? We imagine it begins with an empty belief set and then through dialogue with a knowledgeable entity it expands its belief set (the belief *dynamics* to be outlined below). In one direction of the dialogue information is given to the reasoner while in the other direction the reasoner can be queried as to whether something is a deductive consequence of the reasoner's current knowledge.

We illustrate this process with several dialogues. A prototype reasoner has been implemented using the deductive proof methods described so far. It uses a simple English grammar front-end to receive and impart knowledge, with term equations and inequations as internal representations. The dialogue in the first two columns is the actual interaction with the reasoner, with the third column showing the current belief set whenever it is changed.

With each new fact given in the dialogue, KBI is applied to the current belief set together with the new information to produce a canonical system. This system may simply be $\perp$, indicating that the new set is inconsistent and some process of belief revision is necessary (as discussed in Section 4.2.4). Otherwise the canonical system is kept for use in answering questions (to save recompleting the belief basis).

When a question is asked a proof by contradiction is carried out by adding the negation of the question to the current belief set and applying KBI. If the result is $\perp$ then the simple answer 'Yes' is given. Otherwise the answer is given in terms of the dense set of side conditions returned by KBI. (We will generally use KBI instead of KBC for this purpose, since it involves fewer steps to complete. KBC will only be necessary when the strong uniqueness of its result (see Theorem 3.7) is required, such as for finding a unique set of side conditions).

If there are no non-extraneous side conditions then the non-theorem is in fact an impossible theorem (see Example 3.25) and we give the answer 'No'. If the only non-extraneous side condition is the conclusion then there was no information available to reduce the conclusion and so we answer 'I don't know'. Finally, if we have non-extraneous side conditions distinct from the conclusion then we give the conditions for a theorem as our answer.

EXAMPLE 4.1 (Simple truth). The dialogue in Table 4.1 presents the theorem used at the beginning of Chapter 3. $\diamond$

EXAMPLE 4.2 (Side conditions). The dialogue in Table 4.2 presents a non-theorem, the question not a consequence of the stated beliefs. Here the response is to interpret the side conditions found in the proof attempt. $\diamond$

| Human | Reasoner | Belief Set |
|---|---|---|
| The father of the sister of someone is their father. | Okay. | father sister = father |
| Lucia is the sister of Alice. | Okay. | father sister = father<br>sister Alice = Lucia |
| Is the father of Alice the father of Lucia? | Yes. | |

TABLE 4.1. Dialogue with truth (based on Example 3.1)

| Human | Reasoner | Belief Set |
|---|---|---|
| The father of the sister of someone is their father. | Okay. | father sister = father |
| Lucia is the sister of Alice. | Okay. | father sister = father<br>sister Alice = Lucia |
| Is John the father of Alice? | Only if John is the father of Lucia. | |

TABLE 4.2. Dialogue with side condition (based on Example 3.22)

EXAMPLE 4.3 (No information). In the dialogue of Table 4.3 the reasoner has no information to help answer the question. Of the two side conditions, one is a hypothesis (extraneous) and one is the conclusion. Neither provides a useful answer so the reasoner responds with 'I don't know'. ◇

| Human | Reasoner | Belief Set |
|---|---|---|
| John is male. | Okay. | male John = True |
| Is John the father of Alice. | I don't know. | |

TABLE 4.3. Dialogue without answer

EXAMPLE 4.4 (Impossibility). In Table 4.4 the negation of the question is eliminated by the hypotheses. As seen in Example 3.25, there is thus no way of obtaining a theorem except for contradicting the hypotheses to produce the trivial theorem $\bot \Rightarrow f$. The reasoner thus responds with 'No'.

| Human | Reasoner | Belief Set |
|---|---|---|
| Mary is not male. | Okay. | male Mary $\neq$ True |
| Paul's mother is Mary. | Okay. | male Mary $\neq$ True<br>mother Paul = Mary |
| Is Paul's mother male? | No. | |

TABLE 4.4. Dialogue with impossibility

Compare this dialogue to the one in Table 4.5 where instead of representing 'Mary is not male' by male Mary $\neq$ True we use male Mary $\rightarrow$ False. This time the negated conclusion reduces to True $\neq$ False, an entity inequation that we assume is present in all hypotheses and is thus removed by KBI. We then give the same answer as before.

| Human | Reasoner | Belief Set |
|---|---|---|
| Mary is not male. | Okay. | male Mary = False |
| Paul's mother is Mary. | Okay. | male Mary = False<br>mother Paul = Mary |
| Is Paul's mother male? | No. | |

TABLE 4.5. Dialogue with impossibility (revisited)

The important logical point here is that if True and False were *not* declared to be entities then this second answer would have been the condition 'Only if True is False', while the first answer would still be 'No'. However since we typically do have True and False as entities, we will mostly use equations and rules to express negative information as this is more amenable to rewrite reasoning than an inequation, particularly in the inductive reasoning of Chapter 5.

Related to this is our definition of valuational consistency, a requirement in our proof that inductive theorems were non-grounded. For a system with entities True and False, and the inequation male Mary $\neq$ True, to be valuationally consistent we must also have male Mary = False. $\diamond$

EXAMPLE 4.5 (Implication). The dialogue in Table 4.6 shows how compound sentences (in this case a sentence involving implication) can be used to ask hypothetical questions. The act of proving an implication is equivalent to temporarily adding the antecedent to the hypotheses and then proving the consequent (see Lemma 4.2).

Compare this example to Example 4.9 involving the calculation of meaning, where 'exclusive or' is used instead of 'inclusive or'. $\diamond$

| Human | Reasoner | Belief Set |
|---|---|---|
| The father of the sister of someone is their father. | Okay. | father sister = father |
| John is the father of Lucia. | Okay. | father sister = father<br>father Lucia = John |
| If Lucia is Alice's sister then is John Alice's father? | Yes. | |

TABLE 4.6. Dialogue with hypothetical (based on Example 3.17)

EXAMPLE 4.6 (Local impossibility). In Table 4.7 we have a conjunction that is impossible because one of its components is impossible. Rather than just responding with 'No' we indicate what is causing the impossibility, that is the reduced form of the conclusion that was implied by the hypotheses. $\diamond$

### 4.2.2. Belief Dynamics

The emphasis in the above dialogues is on proof, answering questions from a given (consistent) set of beliefs. In each case new information was easily *believed*, accompanied by a simple update of the belief set. Yet this may not always be the case, and so we need to clarify this process of "believing".

We are actually interested in two kinds of changing belief. The first is direct, where we have a belief set to which we try to add some new belief. Our main concern is the consistency of the new set. If it is consistent then our beliefs can be expanded, otherwise some kind of belief revision is necessary to obtain

| Human | Reasoner | Belief Set |
|---|---|---|
| Mary is not male. | Okay. | male Mary $\neq$ True |
| Mary is the wife of Peter. | Okay. | male Mary $\neq$ True<br>wife Peter = Mary |
| The husband of a person's wife is that person. | Okay. | male Mary $\neq$ True<br>wife Peter = Mary<br>husband wife = i |
| The husband of anyone is male. | Okay. | male Mary $\neq$ True<br>mother Paul = Mary<br>husband wife = i<br>male husband = True ! |
| Are Peter and the wife of Peter both male? | No, because Mary is not male. | |

TABLE 4.7. Dialogue with local impossibility (based on Example 3.28)

a valid belief set in light of the inconsistent information. We shall refer to this process as *deductive belief dynamics*.

In Chapter 5 we introduce an induction algorithm IND which takes a set of ground equations and returns a set of equational conjectures. There we have a functional dependence of our conjectured belief set on the ground observations that we make. Our belief dynamics are thus *empirical*. That is, we are interested in the changes in the conjectured equations as our belief set expands, the changes in our experimental theories as we make new observations. We leave the discussion of such *inductive belief dynamics* to Chapter 6.

### 4.2.3. Contraction

Perhaps the simplest belief dynamic is *belief contraction*, corresponding to some element of $\Gamma$ being removed. For the reasoner, a piece of information is discarded or forgotten. The dialogue in Table 4.8 illustrates one such possibility.

Contraction is a simple process because of the following lemma:

| Human | Reasoner | Belief Set |
|---|---|---|
| John is male. | Okay. | male John = True |
| Is John male? | Yes. | |
| Forget that John is male. | Okay. | $\phi$ |
| Is John male? | I don't know. | |

TABLE 4.8. Dialogue with contraction

LEMMA 4.3 (Hereditary). *If* $\Gamma$ *is consistent and* $\Gamma' \subseteq \Gamma$ *then* $\Gamma'$ *is also consistent.*

PROOF. Anything generated by KBI for $\Gamma'$ is a subset of that created for $\Gamma$ and so if $\Gamma$ did not activate *Contradiction* then neither will $\Gamma'$. □

The hereditary nature of consistency thus ensures that we will still have a belief set.

Contraction may also arise as part of the revision necessary when new information contradicts the existing beliefs, as discussed in the following section.

### 4.2.4. Expansion and Revision

All the dialogues in Section 4.2.1 involved a simple *expansion* of belief with a sentence being added to $\Gamma$. The situation is more complicated if, for example, we already had the belief set $\Gamma = \{\text{father Alice} = \text{John}\}$ and we are told that father Alice $\neq$ John. We cannot add the new information to our knowledge since we lose the consistency required of a belief set.

There are then a number of alternatives as to how we should form our new belief set. At one extreme we have a *closed-minded* approach where we simply reject any new information that contradicts our existing beliefs. This has the obvious advantage of computational expediency but it also has the theoretical advantage of giving monotonic belief change. By never revising our beliefs we are assured that our knowledge at one stage will be a subset of our knowledge at any later stage, based on the assumption that we are learning about a static world. If $\Gamma_f^C$ is the result of a closed-minded revision of $\Gamma$ to accommodate the new sentence $f$ then $\Gamma_f^C = \Gamma$.

At the other extreme we might have faith that information is always improving in some way, and so we insist that our belief set must contain the newly

acquired information. We call this a *narrow-minded* approach, since it is focusing attention on a particular sentence. This is the process assumed in the AGM model of [**20**] and [**24**], and is a standard model for beliefs about a changing world. It will necessarily be non-monotonic and requires additional conditions on the belief revision, such as the principle of minimal change. Here if $\Gamma_f^N$ is the narrow-minded revision of $\Gamma$ to accommodate $f$ then $\Gamma_f^N \neq \Gamma_f^C$, since $f \notin \Gamma_f^C$.

Between these two extremes we may take the *broad-minded* approach where we treat our current belief set and the new information as a pool of knowledge from which we must extract a consistent subset. Again this may not be monotonic and again it requires conditions for making a rational choice of the subset. For instance we would not want to choose the empty subset as our beliefs, even though it is always consistent.

This broad-minded approach will return in Chapter 5 where it is the only one applicable to finding a consistent theory from a set of conjectures. There we similarly treat the conjectures as a pool of knowledge from which we want to obtain such a theory. We cannot single out a single conjecture which we want to include or exclude since all are contingent on observations. The problem is somewhat different though since the consistency of a theory is defined relative to the given database of observations. However, many of our ideas here will be applicable to the inductive case, especially the notion of a *maximally* consistent set.

In the case of belief revision, whether it be narrow-minded or broad-minded, an obvious criterion for what we believe is that it should be maximal. We should not limit our beliefs to a particular subset when a further piece of knowledge can be consistently added to the subset. This is captured in the following definition:

DEFINITION 4.2 (Maximal Consistency). If $F \subseteq \mathcal{F}$ then $C \subseteq F$ is a *maximally consistent* subset of $F$ if $C$ is consistent and there is no other $f \in F \backslash C$ such that $C \cup f$ is consistent.

This maximality requirement may be too eager in some cases. If we have an inconsistency caused by two equations

$$\mathsf{father\ Alice} = \mathsf{John}, \quad \mathsf{father\ Alice} = \mathsf{Paul},$$

then it may be preferable to remove belief in both equations, even if we could consistently believe in one of them. By taking a maximal set we are essentially adopting a *working system*, a system from which we can make deductions but which is open to clarification through later revision. We illustrate this in Example 4.7.

A system $F$ need not usually have a unique maximally consistent subset, so this does not give a sufficient condition for choosing a new belief set. One

case where we do have uniqueness is as follows. Let $\Gamma_f^B$ be the result of a broad-minded revision of $\Gamma$ to accommodate the sentence $f$. Then $f \notin \Gamma_f^B$ if and only if $\Gamma_f^B = \Gamma$, and hence $\Gamma_f^B$ is unique. Equivalently, $f \notin \Gamma_f^B$ if and only if $\Gamma_f^B = \Gamma_f^C$.

If we are faced with multiple maximal sets then some further criteria are required to determine the new belief basis. Here our *selection function* [**20**] will be to take an arbitrary maximal set. In Chapter 5, where conjectured beliefs are based on observations, we can instead base our choice on the predictions each maximal set makes about future observations.

EXAMPLE 4.7 (Working system). The dialogue of Table 4.9 illustrates how a maximal subset can be thought of as a working system, a collection of basic beliefs which is used for reasoning but which may subsequently be revised. With the third statement the basic beliefs are inconsistent and the reasoner must chose a maximal system from them. This is maintained until the final statement when new inconsistency requires a further revision. ◇

| **Human** | **Reasoner** | **Belief Set** |
|---|---|---|
| The husband of a person's. mother is their father. | Okay. | husband mother = father |
| Alice's father is John. | Okay. | husband mother = father<br>father Alice = John |
| Alice's father is Paul. | Okay. | husband mother = father<br>father Alice = Paul |
| Alice's mother is Jill. | Okay. | husband mother = father<br>father Alice = Paul<br>mother Alice = Jill |
| Jill's husband is John. | Okay. | husband mother = father<br>father Alice = John<br>mother Alice = Jill<br>husband Jill = John |

TABLE 4.9. Dialogue with a changing view

A simple illustration of the closed-minded approach is given in Example 7.2.

Sattar and Ghose [45] have looked at the notion of maximally consistent systems for belief revision and the generation of experiments to resolve the conflict between multiple maximal systems. Such *deductive experiments* arise from the addition of new information, whereas the *abductive experiments*, related to the side conditions of Section 3.5, were the result of questions being asked of the reasoner. Our principle interest in experiments will come in Chapter 6 where we look at *inductive experiments*. There we have potential conflict between conjectures from a given set of data, and seek to suggest experiments whose results will most likely refute false conjectures.

## 4.3. Internal Truth: The Meaning of Conjunction

Until now we have represented a set of beliefs as just that, a set of equations and inequations from $T_\Sigma$. However there are structures present in $T_\Sigma$ which allow us to discuss belief *internally*, without reference to sets of things. Specifically, the belief set

$$\{e_1, e_2, \ldots, e_n\}$$

can be expressed as the single term

$$\mathsf{and}\langle e_1, e_2, \ldots, e_n\rangle.$$

Here $\mathsf{and} : \mathsf{sentence} \longrightarrow \mathsf{sentence}$ is the (associative and commutative) boolean 'and' function, in contrast to the external connective $\wedge$. We will usually write $\mathsf{and}\langle e_1, e_2, \ldots, e_n\rangle$ as the juxtaposition $e_1 e_2 \cdots e_n$.

For internal truth we no longer have a dynamic belief set from which we attempt to prove equations or inequations. Instead we look at the reduction of a single expression to normal form, where our beliefs are now part of the expression being reduced. Rather than giving a dialogue, an implementation of this internal truth is a calculator.

### 4.3.1. Meaning

Accounts of meaning and its logic are given in [16] and [17], which is an overview of the Definitional Reasoning system. For our purposes we view the *meaning* of a term in $T_\Sigma$ to be its normal form with respect to a confluent and terminating set of rules. Thus meaning can be viewed as a function $\mathcal{M} : T_\Sigma \longrightarrow T_\Sigma$ from terms to terms. We may then say that two terms $s$ and $t$ *have the same meaning* if simply $\mathcal{M}(s) \equiv \mathcal{M}(t)$, where $\equiv$ is term (string) equality. This use of the function $\mathcal{M}$ can be seen as a generalization of using Knuth-Bendix and rewrite reduction to show equality in an equational system.

Here the known rules are the structural rules given in Section 2.3, together with the conjunction rule and propositional rules to be given below. Additional rules may be added to these, as detailed in [17]. This then gives

a blending of external and internal truth, the external definitions refining knowledge by giving more and more terms the same meaning.

The main burden of reasoning in a meaning-based system is placed on the processing of conjunction. The normal form of a conjunction of equations is the result of applying KBC to the corresponding equational system. That is,

$$\mathcal{M}(e_1 e_2 \cdots e_n) = \mathrm{KBC}(\{e_1, e_2, \ldots, e_n\})$$

where each $e_j$ has head $=_\sigma$ or $\neq_\sigma$, and where we replace $\rightarrow$ everywhere in the result of KBC by $=$ and write the set as a conjunction. If KBC gives $\perp$ then we say the meaning of the conjunction is False. KBC is used to ensure a given internal expression has a unique meaning.

EXAMPLE 4.8 (Meanings of conjunctions). The above definitions give

$$\begin{aligned} & \mathcal{M}[(\mathsf{male\ father} = \mathsf{True}\ !)(\mathsf{father\ Alice} = \mathsf{John})] \\ \rightarrow\ & (\mathsf{male\ father} = \mathsf{True}\ !)(\mathsf{father\ Alice} = \mathsf{John})(\mathsf{male\ John} = \mathsf{True}), \end{aligned}$$

while

$$\mathcal{M}[\mathsf{father\ Alice} = \mathsf{John})(\mathsf{father\ Alice} \neq \mathsf{John})] \rightarrow \mathsf{False}.$$

◇

To capture additional connectives we also require the boolean addition xor : sentence $\rightarrow$ sentence (*exclusive disjunction*). Again xor is associative and commutative, and we will write it in infix notation as $+$. Any meaning function $\mathcal{M}$ must include

$$\psi + \psi \rightarrow \mathsf{False},$$
$$\mathsf{False} + \psi \rightarrow \psi,$$

for any $\psi$. We then define the meaning of other operators by

$$\mathsf{not}\ \psi \rightarrow \psi + \mathsf{True},$$
$$\mathsf{implies}\ \langle \psi, \theta \rangle \rightarrow \psi\theta + \psi + \mathsf{True},$$
$$\mathsf{or}\ \langle \psi, \theta \rangle \rightarrow \psi + \theta + \psi\theta.$$

These definitions give the rich logical structure of meaning, outlined in [**16**]. Our main interest here is in the implies operator. If we want to show a conjunction $F$ logically implies an $f \in \mathcal{F}$, we need to show that $\mathcal{M}[Ff + F + \mathsf{True}] = \mathsf{True}$. Thus, from our rule for xor, we need to show $\mathcal{M}[Ff] = \mathcal{M}[F]$. That is, we must show $\mathrm{KBC}(F \cup f) = \mathrm{KBC}(F)$, which is exactly the process of proof by invariance. We can summarize this by the following consequence of Theorem 3.7:

COROLLARY 4.2. $\mathcal{M}[\mathsf{implies}\langle F, f \rangle] = \mathsf{True}$ *if and only if* $F \Rightarrow f$.

Hence proof by invariance plays the central deductive role for internalized truth that proof by contradiction has played for our external truth. We will thus use the infix notation $\Rightarrow$ for the operator implies.

EXAMPLE 4.9 (Internal proof). The following calculation evaluates the meaning of the theorem in Example 3.15, when male < father:

$$
\begin{aligned}
& \mathcal{M}[F \Rightarrow f] \\
\rightarrow\ & \mathcal{M}[(\mathsf{male\ father} = \mathsf{True}\ !)(\mathsf{male\ Alice} \neq \mathsf{True}) \Rightarrow (\mathsf{father\ John} \neq \mathsf{Alice})] \\
\rightarrow\ & \mathcal{M}[(\mathsf{male\ father} = \mathsf{True}\ !)(\mathsf{male\ Alice} \neq \mathsf{True})(\mathsf{father\ John} \neq \mathsf{Alice}) + \\
& (\mathsf{male\ father} = \mathsf{True}\ !)(\mathsf{male\ Alice} \neq \mathsf{True}) + \mathsf{True}] \\
\rightarrow\ & \mathcal{M}[(\mathsf{male\ father} = \mathsf{True}\ !)(\mathsf{male\ Alice} \neq \mathsf{True}) + \\
& (\mathsf{male\ father} = \mathsf{True}\ !)(\mathsf{male\ Alice} \neq \mathsf{True}) + \mathsf{True}] \\
\rightarrow\ & \mathcal{M}[\mathsf{False} + \mathsf{True}] \\
\rightarrow\ & \mathcal{M}[\mathsf{True}] \\
\rightarrow\ & \mathsf{True}
\end{aligned}
$$

◇

CHAPTER 5

# Learning Equations from a Database

So far we have discussed a rewriting-based framework for *deductive* reasoning. For example, from the equational system

$$\mathsf{father\ Alice} \to \mathsf{John}$$
$$\mathsf{male\ father} \to \mathsf{True}\ !$$

we are able to deduce that $\mathsf{male\ John} = \mathsf{True}$. In this section we give a procedure for *inductive* reasoning, so that from

$$\mathsf{father\ Alice} \to \mathsf{John}$$
$$\mathsf{male\ John} \to \mathsf{True}$$

we may conjecture that $\mathsf{male\ father} = \mathsf{True}\ !$. We have evidence for this new fact since whenever we apply the function **male father** to a person we get the same result as applying **True** !. Of course this raises the immediate question as to how much belief we should put in our generalization. Making a conjecture and asking such questions is the basis of scientific inquiry. We will come to a discussion of belief, in Chapter 6, after first presenting the algorithm in detail.

## 5.1. Induction Method

Generalizing the above example, we would conjecture that $f = g$ if we find that $fx = gx$ for all appropriate $x$. This principle will be the basis for database induction.

### 5.1.1. Facts and Conjectures

The induction procedure IND takes a finite, complete and inductively consistent set of rewrite rules $\Delta$ (the *database*) and returns a set of equations (the *conjectures*). (Note that $\Delta$ must be consistent as otherwise it's completion would be $\perp$). The equations in the result will be of two kinds, facts and conjectures.

DEFINITION 5.1 (Fact). A *fact* is a function equation $f = g$ such that $\Delta \not\Rightarrow f = g$ and $fa =_\Delta ga$ for all $\Sigma$-entities $a \in \mathrm{dom}(f)$.

That is, a fact is a function equation which holds when applied to any entity of appropriate type.

DEFINITION 5.2 (Conjecture). A *conjecture* is a function equation $f \simeq g$ such that $\Delta \not\Rightarrow (f = g)$, $\Delta \not\Rightarrow (f \neq g)$, $\Delta \Rightarrow (fa = ga)$ for at least one $\Sigma$-entity $a \in \mathrm{dom}(f)$ and there is no $\Sigma$-entity $b \in dom(f)$ such that $\Delta \Rightarrow (fb \neq gb)$.

A *conjecture* is an incomplete fact, a function equation which holds for at least one grounded word and is not falsified by any other grounded word. A conjecture is not a fact because information is absent in $\Delta$ about the meaning of the composition of one of the functions and some entity. We use the notation $\simeq$ to indicate this lack of universal support. However, note that a fact is immediately a conjecture, and we will often refer to facts and conjectures collectively as simply conjectures.

This illustrates well why we don't adopt a closed-world assumption. We instead take the scientific view that the truth of some equations may simply be unknown. Truth or falsity, if not the deductive consequence of a belief basis, can only be established by carrying out new experiments.

The condition that $\Delta \not\Rightarrow (f \neq g)$ ensures that the conjectured $f \simeq g$ is not inconsistent with the information in $\Delta$. This constraint is not necessary for facts, as seen in Lemma 5.1 below.

From these definitions it is clear that facts can never produce new database information, capturing only truth that is already present. In this sense, a fact represents *summative* induction, the equation summarizing the complete information we have about the data. Although a fact can generate no new data, it can be used to compress the database by eliminating data rules which are implied by it (see Section 6.4).

If a conjecture is accepted as a fact, it can then produce new database information by essentially filling in the gaps which prevented it from being called a fact in the first place. A conjecture is a form of *ampliative* induction, the equation providing new information and thus amplifying our knowledge [**8**]. Conjectures will be our main focus here as they have close and interesting parallels with scientific method.

LEMMA 5.1. *A fact from $\Delta$ is consistent with $\Delta$.*

PROOF. If a fact $f = g$ is not consistent with $\Delta$ then there must be an inequation $h_1 \neq h_2$ in $\Delta$ such that $h_1 =_{\Delta \cup \{f=g\}} h_2$. As in the proof of Theorem 3.6, we can use the localized functional completeness of $f = g$ and the inductive consistency of $\Delta$ to show that $\Delta$ is inconsistent, contradicting our assumption for $\Delta$. Hence $f = g$ is consistent with $\Delta$. □

The following two results give a concrete relationship between this inductive process and the inductive theorems of Section 3.3, that is, between scientific and mathematical induction.

THEOREM 5.1. *A fact from $\Delta$ is an inductive theorem of $\Delta$.*

This theorem, together with its proof, is similar to our earlier characterization of unambiguous systems in terms of functional completeness. However we do not need an unambiguous system to have a theorem in the sense of Definition 3.11. Theorem 5.1 shows that a local form of functional completeness, embodied in the definition of fact, is sufficient for an equation to be a theorem. This also justifies writing the equality $f = g$ for a fact.

PROOF. Since any fact $f = g$ is consistent with $\Delta$, by Lemma 5.1, there must be at least one maximally consistent theory of $\Delta$ containing $f = g$. Recall that a theorem of $\Delta$ is an equation in the intersection of all maximally consistent theories of $\Delta$. Suppose then that $f = g$ is not a theorem so that there is some maximally consistent theory, $C$, of $\Delta$ not containing $f = g$. Thus there is some inequation $h_1 \neq h_2$ in $\Delta$ such that $h_1 =_{C \cup \{f=g\}} h_2$. Again we can follow the proof of Theorem 3.6 to show that $C$ must be inconsistent, proving that $f = g$ is a theorem of $\Delta$.

That $f = g$ is inductive, rather than deductive, is simply part of the definition of a fact. □

The converse of this result, that inductive theorems are facts, holds similarly provided that $\Delta$ is valuationally consistent (see Theorem 3.4).

THEOREM 5.2. *If $\Delta$ gives rise to a conjecture then $\Delta$ is ambiguous.*

PROOF. Suppose $f \simeq g$ is a conjecture from $\Delta$. Since $f = g$ is not a fact, for some entity $a \in \mathrm{dom}(f)$ there is either no entity $b \in \mathrm{cod}(f)$ such that $fa =_\Delta b$ or no entity $c \in \mathrm{cod}(g)$ such that $ga =_\Delta c$. In either case $\Delta$ cannot be functionally complete and so by Theorem 3.5 it is ambiguous. □

This second result is perhaps our "fundamental theorem". If a system is ambiguous then we are unable to prove an inductive theorem by consistency (Theorem 3.2). Thus a conjecture can be simply characterized as an inductive theorem that cannot be proved.

### 5.1.2. Induction Algorithm

Let $\Sigma_\sigma$ denote the set of all $\Sigma$-entities of type $\sigma$ and let $\Sigma^n_{\sigma \to \tau}$ denote the set of all $\Sigma$-functions with domain $\sigma$ and length at most $n$.

For a function term $f$, an ordered set of words $W = \{w_i; i \in I\}$, and a given rewrite system $\Delta$, define $f(W) \downarrow_\Delta$ to be the ordered set $\{n_i; i \in I\}$, with

each $n_i$ defined as follows:

$$n_i = \begin{cases} s & \text{if } fw_i \rightarrow_\Delta^* s \text{ for some } \Sigma\text{-entity } s \\ * & \text{otherwise} \end{cases}$$

The condition that $s$ be an entity in the first case will correspond to the requirement that we must have complete knowledge about the function value before we make any conjecture. The special term $*$ in the second case indicates that no complete information is present in $\Delta$ about the value of the function for that particular word.

Finally, for ordered subsets $S_1, S_2$ of $T_\Sigma \cup \{*\}$ having equal length, we say $S_1 = S_2$ if $S_1$ and $S_2$ are identical at each position and both are free of the element $*$. We write $S_1 \simeq S_2$ if $S_1$ and $S_2$ are identical at each position where neither has a $*$.

With these definitions, we can now present the algorithm:

procedure IND$(\Delta, n)$
  $E_{\Delta,n} := \phi$
  for each sort $\sigma$
    for each $f, g \in \Sigma^n_{\sigma\rightarrow\tau}\downarrow_\Delta$, $f \not\equiv g$,
      if $f(\Sigma_\sigma)\downarrow_\Delta = g(\Sigma_\sigma)\downarrow_\Delta$ then $E_{\Delta,n} := E_{\Delta,n} \cup \{f = g\}$
      if $f(\Sigma_\sigma)\downarrow_\Delta \simeq g(\Sigma_\sigma)\downarrow_\Delta$ then $E_{\Delta,n} := E_{\Delta,n} \cup \{f \simeq g\}$
  IND $:= E_{\Delta,n}$
end.

For each $\Sigma$-sort $\sigma$ we look at the set of $\Sigma$-functions with domain $\sigma$, reduced by $\Delta$. We take pairs of distinct $f, g$ from this set and apply each of them to an ordered list of all grounded words with codomain $\sigma$. These applications are then reduced by $\Delta$, using $*$ for any whose normal form is not an entity. If the reduced lists are free of $*$ and identical then we add $f = g$ as a fact. If they are identical at each element where neither has $*$, and there is at least one such element, then we add $f = g$ as a conjecture (which we write as $f \simeq g$).

In an actual implementation of the procedure there are many efficiency improvements that can be made to reduce the number of functions to be considered and the number of normal form reductions to be performed. In general though, the number of functions to be considered grows exponentially with $n$. To work with this we apply the induction algorithm iteratively. Starting with $\Delta_1 = \text{IND}(\Delta, 1)$, we evaluate

$$\Delta_{j+1} = \Delta_j \cup \text{IND}(\Delta \cup \Delta_j, j+1),$$

the conjectures arising from the result of induction for length $j$ being used to reduce the candidate functions of length $j+1$. We then say that the result of the induction of $\Delta$ is the set $\Delta_\infty$.

We would like to show that $\Delta_\infty$ is finite, so that a finite set of observations can only support a finite number of conjectures. Let $F$ and $G$ be the sets of $\Sigma$-functions and grounded $\Sigma$-words, respectively. Let $V(G)$ be the set of vectors of length $|G|$ with components drawn from $G \cup \{*\}$, and let $v_0 \in V(G)$ be the initial vector $G$ with elements in some arbitrary order. Note that for a finite $\Delta$, the set $G$ is finite and thus $V(G)$ is finite, with $|V(G)| = (|G|+1)^{|G|}$.

Suppose we have a sequence $f_1, f_2, \ldots$ of functions from $F$. Each $f_j$ gives a map from $V(G)$ to $V(G)$ define by

$$f_j(\langle \alpha_1, \ldots, \alpha_{|G|} \rangle) = \langle f_j(\alpha_1), \ldots, f_j(\alpha_{|G|}) \rangle$$

where $f_j(\alpha_i)$ is the $\Delta$-normal form of $f_j \alpha_i$, taking value $*$ if the normal form is not a word and with $f_j * \rightarrow *$. Hence each $f_j$ maps a finite set into itself and so the sequence

$$v_0, f^1 v_0, f^2 v_0, \ldots,$$

where $f^j = f_j f_{j-1} \cdots f_1$, must eventually cycle. That is, for some $K, h$,

$$f^k v_0 = f^{k+h} v_0$$

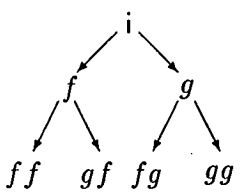
for all $k \geq K$. Hence any function appearing beyond $f^{K+h}$ has the same image as a previous function, with equivalence identified by reducing with $f^{K+h} \rightarrow f^K$. This proves the following lemma:

LEMMA 5.2. *Let $F$ be the set of function words in a database $R$. Then for any sequence of functions $f_1, f_2, \ldots$ from $F$ there are only finitely many distinct functions $f^j$, where $f^j = f^j f^{j-1} \cdots f_1$.*

This lemma is the basis of the following important result.

THEOREM 5.3 (Termination of Induction). *For a given database $\Delta$ there exists some $K$ such that $\Delta_k = \Delta_K$ for all $k > K$.*

PROOF. Every possible function that be built from $m$ function words is a node of the tree with (polymorphic) root node i and where each node branches into at most $m$ child nodes, each child formed by composing one of the function words with the parent node, if types allow. The first two steps of this branching process with only two function words, $f, g$, is shown in the following figure.



Each path in the tree is simply a sequence of function compositions, so by Lemma 5.2 when applied to an initial vector a cycle is found in the images of the functions. The conjecture will then be formed between the first functions with the same image, after which subsequent functions in the path can be reduced to earlier functions with that conjecture. Hence the reduced set of functions produced by that path is finite.

This argument applies to all paths in the tree, so that the total reduced set of functions which could be considered for induction is finite. So by using previous conjectures to reduce new candidate functions, eventually the induction process will return nothing new. □

Hence from a finite amount of knowledge we can only make a finite number of conjectures using inductive reasoning alone. This is an interesting distinction to make in comparison to deductive reasoning where finite knowledge can produce infinite numbers of deductive consequences. Example 5.3 gives an example of the finite and infinite interaction of inductive and deductive reasoning.

Even though this theorem shows the inductive knowledge will be finite, the number of conjectures of general length can be quite large. This is especially so early on in a series of experiments when many 'wild' conjectures, conjectures at odds with the modelled world, often appear before being refuted by further observations. In most cases we will concentrate on finding $\mathrm{IND}(\Delta, 2)$ only, conjecturing simple relationships between functions. In essence, this is Occam's razor at work. A single function is used to model a *process* which is conceptually basic (such as father and male). Terms of greater length correspond to more and more complex compound processes (such as male father father). By starting with $\mathrm{IND}(\Delta, 2)$ we are looking initially for simple explanations of the data and only if we cannot find any do we search for more complex theories.

### 5.1.3. Families and Gender

For this section we use the following signature:

$$\Sigma = \left\{ \begin{array}{l} \mathsf{True}, \mathsf{False} \in \mathsf{sentence}, \\ \mathsf{John}, \mathsf{Peter}, \mathsf{Alice}, \mathsf{George}, \mathsf{Jill}, \mathsf{Paul} \in \mathsf{person}, \\ \mathsf{not} : \mathsf{sentence} \to \mathsf{sentence}, \\ \mathsf{male}, \mathsf{female} : \mathsf{person} \to \mathsf{sentence}, \\ \mathsf{father}, \mathsf{mother}, \mathsf{fatherinlaw} : \mathsf{person} \to \mathsf{person} \end{array} \right\}$$

EXAMPLE 5.1 (Refinement of conjectures). Consider the following database consisting of four rules:

$$\Delta = \left\{ \begin{array}{l} \mathsf{father\ Paul} \to \mathsf{Peter},\ \mathsf{father\ Alice} \to \mathsf{John}, \\ \mathsf{male\ Peter} \to \mathsf{True},\ \mathsf{male\ John} \to \mathsf{True} \end{array} \right\}.$$

Applying IND to $\Delta$ gives the two conjectures

$$\Delta_2 = \mathrm{IND}(\Delta, 2) = \{\mathsf{male} \simeq \mathsf{True}\ !,\ \mathsf{male\ father} \simeq \mathsf{True}\ !\}.$$

The first of these is false in the world we want to model, while the second is true. The reason the first appears is simply that the only people whose gender is mentioned in the database are male. There are no counter-examples to refute the conjecture. As the breadth of data increases, fewer such false conjectures will be made. Consider a larger database

$$\Delta' = \Delta \cup \{\mathsf{male\ Alice} \rightarrow \mathsf{False}\}$$

where now

$$\Delta'_2 = \mathrm{IND}(\Delta', 2) = \{\mathsf{male\ father} \simeq \mathsf{True}\ !\}.$$

Unless false data is included in the database, this remaining conjecture will persist, until complete information is given for the left and right functions to make it a fact. In the meantime we must decide what to do with the conjecture. If we assume it is true then we can deduce, for instance, that Peter's father is male, even though we have no information about who he is. But this simple idea is also dangerous - using the original $\Delta$ we would have similarly concluded that Alice was male. $\diamond$

This process is similar to the process of making conjectures in science. The conjecture, and all of its deductive consequences, are assumed true until a counter-example is found. A counter-example may well be present in the existing data. Although a single conjecture alone can never produce a contradiction, since it would then have never been conjectured, combinations of conjectures can produce new information which contradicts the known data. Consider the following example.

EXAMPLE 5.2 (Consistency of conjectures). With the database

$$\Delta = \left\{ \begin{array}{l} \mathsf{male\ Paul} \rightarrow \mathsf{True},\ \mathsf{male\ Alice} \rightarrow \mathsf{False}, \\ \mathsf{female\ Paul} \rightarrow \mathsf{False}, \\ \mathsf{not\ False} \rightarrow \mathsf{True},\ \mathsf{not\ True} \rightarrow \mathsf{False} \end{array} \right\}$$

we obtain the conjectures

$$\Delta_2 = \mathrm{IND}(\Delta, 2) = \left\{ \begin{array}{l} \mathsf{female} \simeq \mathsf{False}\ !,\ \mathsf{not\ female} \simeq \mathsf{True}\ !, \\ \mathsf{female} \simeq \mathsf{not\ male},\ \mathsf{not\ female} \simeq \mathsf{male}, \\ \mathsf{not\ not} = \mathsf{i} \end{array} \right\}.$$

The first two conjectures are not true in the world we are modelling but are consistent with the given data. However, by combining the second and fourth conjectures we are able to deduce the new conjecture $\mathsf{male} \simeq \mathsf{True}\ !$, which then implies that $\mathsf{male\ Alice} \rightarrow \mathsf{True}$, contradicting the information in $\Delta$. We discuss this issue of the *consistency* of a system of conjectures with respect to the original data in section 6.2. $\diamond$

EXAMPLE 5.3 (Finite induction and infinite deduction). Even though Theorem 5.3 guarantees that induction can only produce a finite number of conjectured equations, the following example illustrates the 'amount' of new information these equations can give. Consider a fresh database

$$\Delta = \left\{ \begin{array}{l} \mathsf{father\ Alice} \rightarrow \mathsf{John},\ \mathsf{father\ Jill} \rightarrow \mathsf{George}, \\ \mathsf{mother\ Alice} \rightarrow \mathsf{Jill},\ \mathsf{fatherinlaw\ John} \rightarrow \mathsf{George} \end{array} \right\}.$$

We find

$$\Delta_2 = \mathrm{IND}(\Delta, 2) = \{\mathsf{fatherinlaw\ father} \simeq \mathsf{father\ mother}\}.$$

As seen in Example 3.5, applying KB to $\Delta \cup \Delta_2$ gives a rewrite system containing the rules

$$\mathsf{father\ mother}^k\ \mathsf{Jill} \rightarrow \mathsf{fatherinlaw}^k\ \mathsf{George}$$

for any $k \geq 1$. Thus the induction process applied to a *finite* database may result in a system with an *infinite* number of deductive consequences. ◇

EXAMPLE 5.4 (Recursive conjectures). Consider

$$\Delta = \left\{ \begin{array}{l} \mathsf{parent\ Alice} \rightarrow \{\mathsf{Jill}, \mathsf{John}\}, \\ \mathsf{ancestor\ Alice} \rightarrow \{\mathsf{George}, \mathsf{Henry}, \mathsf{Jill}, \mathsf{John}, \mathsf{Kate}, \mathsf{Lucia}\}, \\ \mathsf{ancestor\ Jill} \rightarrow \{\mathsf{George}, \mathsf{Lucia}\}, \\ \mathsf{ancestor\ John} \rightarrow \{\mathsf{Henry}, \mathsf{Kate}\} \end{array} \right\}.$$

Induction gives the conjectures

$$\Delta_2 = \mathrm{IND}(\Delta, 2) = \left\{ \begin{array}{l} \mathsf{ancestor} \simeq \{\mathsf{ancestor}, \mathsf{parent}\}, \\ \mathsf{ancestor} \simeq \{\mathsf{ancestor}, \mathsf{ancestor\ parent}\}, \\ \mathsf{ancestor} \simeq \{\mathsf{parent}, \mathsf{ancestor\ parent}\} \end{array} \right\}.$$

The first two capture the subset property that **parent** $\subseteq$ **ancestor** and the recursive subset property that **ancestor** $\subseteq$ **ancestor parent**. The remaining conjecture is the recursive definition of **ancestor** in terms of **parent**, where someone's ancestor is either a parent or an ancestor of a parent. (Note that this 'or' is not necessarily exclusive since we are using sets). ◇

EXAMPLE 5.5 (Facts from infinite domains). As described in the comments following Example 3.13, we cannot induce any facts involving functions of type **person** $\rightarrow$ **person** because to do so would require an infinite regression of information about the function's values. For example, we can never obtain the fact **male father** = **True** ! because if we knew Alice's father was John and that John was male, then we would need to know who John's father was, and so on.

We described two approaches for escaping this situation when trying to prove inductive theorems (corresponding here to facts). The Peano scheme is not very appropriate for the scientific model of making observations and conjecturing information from them, so we focus here on the use of the addition of the special element $\omega_\sigma$.

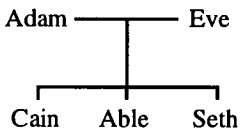



FIGURE 5.1. Adam and Eve's family tree

Consider the well-known family tree in Figure 5.1. Based on it we make the following observations:

$$\Delta = \left\{ \begin{array}{l} \mathsf{mother\ Seth} \rightarrow \mathsf{Eve},\ \mathsf{male\ Seth} \rightarrow \mathsf{False}, \\ \mathsf{male\ Eve} \rightarrow \mathsf{False},\ \mathsf{mother\ Eve} \rightarrow \omega \end{array} \right\}.$$

The last rule captures the observation that Eve has no mother, or that **mother Eve** is unknowable. From $\Delta$ we know the value of **male mother** for all elements of **person** in the signature

$$\Sigma = \left\{ \begin{array}{l} \mathsf{Eve}, \mathsf{Seth} \in \mathsf{person}, \\ \mathsf{mother} : \mathsf{person} \rightarrow \mathsf{person}, \\ \mathsf{male} : \mathsf{person} \rightarrow \mathsf{sentence} \end{array} \right\}.$$

Hence induction gives the single *fact*

$$\mathrm{IND}(\Delta, 2) = \{\mathsf{male\ mother} = \mathsf{False}\ !\},$$

Indeed it is perhaps this ability, of replacing conjectures by fact, that makes religion so appealing! $\diamond$

Our use of $\omega$ is quite similar to the 'special out of world constant' proposed for FOIL in [**42**]. There the concern is to eliminate spurious inductive inferences caused by the closed world assumption. Anything that is not positively stated in the database is assumed to be false. The standard example is to give FOIL a database about the reverse and concatenation relations for all lists of maximum length three. FOIL then inductively generates the clause that a list is its own reverse if it can be concatenated with itself to give another list, which is certainly true for the given data.

In our rewrite induction we do not use the closed world assumption. Such an example would simply be taken as a conjecture (part of a working theory) and maintained until it was falsified or it became a fact by being completely tested on all elements in its domain. Of course for lists, as with the sort **person**, we cannot obtain a fact because there are always lists of size one larger to apply the conjecture to. This is where our interest in the element $\omega$ lies, 'completing' a database so that we may *prove* a conjecture in the sense of Section 3.3.

However the use of $\omega$ is artificial and not necessarily desirable since in the standard scientific induction that we are modelling facts rarely exist. Even though the sun has been recorded to rise on many days there is no certainty

that it will rise tomorrow. Furthermore, as summative knowledge, a fact cannot provide us with any predictions (see Lemma 6.4) and so in general has less practical value for us than a conjecture.

### 5.1.4. Inductive Generation of Function Inequations

The inductive process can be seen as the inverse of the quantification description given in Section 4.1.1. An inequation represents existential information, rather than the universal information of a function equation, and so to induce an inequation $f \neq g$ we need only make one observation where $fa \neq ga$.

However, we saw in Section 3.4.6 that this is a trivial kind of induction because the conjectured inequation $f \neq g$ is in fact a deductive consequence of the database, and hence does not conform to our definitions of conjecture and fact. Indeed simply making the observation $fa \neq gb$ will implicitly imply $f \neq g$ in any future proof by contradiction, so explicitly adding it as conjectured knowledge is unnecessary.

## 5.2. Language Dynamics

As with deductive reasoning, we also have belief dynamics for our inductive process. As the database is extended, new conjectures may be made and old conjectures may be falsified. In later sections we will look at what we can say about these changes in the inductive beliefs, both through deductive measures of their consistency and through statistical assumptions about the world being modelled by the database.

However, we saw in Section 3.3 that inductive theorems are also very dependent on the underlying language generated by the signature $\Sigma$. Suppose we have a deductive theorem $F \Rightarrow f$, with $F \subseteq \mathcal{F}_\Sigma$ for some $\Sigma$. If we extend the signature to $\Sigma'$, so that $\mathcal{F}_\Sigma \subseteq \mathcal{F}_{\Sigma'}$ and we now view $F \subseteq \mathcal{F}_{\Sigma'}$, then we will still have $F \Rightarrow f$. Yet for a similar inductive theorem the extension to $\Sigma'$ may destroy the functional completeness needed for its proof. For example, the system $F = \{\mathsf{male\ John} = \mathsf{True}\}$ over a signature with single entity $\mathsf{John} \in \mathsf{Person}$ gives rise to the inductive theorem $\mathsf{male} = \mathsf{True}\ !$. However, if we extend the signature by adding $\mathsf{Paul} \in \mathsf{Person}$ then, since we do not know the value of $\mathsf{male\ Paul}$ in $F$, $\mathsf{male} = \mathsf{True}\ !$ is no longer an inductive theorem of $F$.

Correspondingly, if $f = g$ is a fact from a database $\Delta \subseteq \mathcal{F}_\Sigma$ then $f = g$ will not necessarily be a fact from $\Delta \subseteq \mathcal{F}_{\Sigma'}$. Additional entities in the domain of $f = g$ will reduce it to a conjecture, with further extensions to $\Delta$ possibly falsifying it. The next result gives the obvious condition for ensuring $f = g$ remains a fact.

LEMMA 5.3 (Language Extension). *Suppose $f = g$ is a fact from some database $\Delta \subseteq \mathcal{F}_\Sigma$ and that $\Delta' \subseteq \mathcal{F}_{\Sigma'}$ is an extension of $\Delta$ such that $\Delta \subseteq \Delta'$ and $\mathcal{F}_\Sigma \subseteq \mathcal{F}_{\Sigma'}$. Then $f = g$ is a fact for $\Delta'$ if $\Delta' \not\Rightarrow (f = g)$ and $\Sigma_{dom(f=g)} = \Sigma'_{dom(f=g)}$.*

PROOF. For all $\Sigma$-entities $a \in \mathrm{dom}(f)$, and hence for all $\Sigma'$-entities $a \in \mathrm{dom}(f)$, $fa =_\Delta ga$. Since $\Delta \subseteq \Delta'$, by the monotonicity of $\Rightarrow$ (Theorem 4.1) we must also have $fa =_{\Delta'} ga$. Thus $f = g$ is a fact from $\Delta'$. □

The requirement $\Delta' \not\Rightarrow (f = g)$ will automatically be satisfied if $\Delta'$ consists only of grounded equations.

This simple result captures a basic premise of scientific enquiry. For instance, if we have an inductive theorem about the natural numbers and then turn to make observations about the motions of planets, we are guaranteed that our number theorem will still hold. Extending our language in a distinct direction cannot effect known theorems. As noted above, this is trivially true for deductive theorems, and now we see it is true for inductive theorems as well.

The converse of the result need not hold. Often the addition of a new entity is accompanied by the addition of data about the application of functions to the entity. This will maintain functional completeness and so give a fact (if it is not falsified), even though $\Sigma_{\mathrm{dom}(f=g)} \neq \Sigma'_{\mathrm{dom}(f=g)}$. However the converse will hold whenever $\Delta = \Delta'$.

We have a similar, but weaker, result for conjectures.

LEMMA 5.4. *Suppose $f = g$ is a conjecture from some database $\Delta \subseteq \mathcal{F}_\Sigma$ and that $\Sigma'$ is such that $\mathcal{F}_\Sigma \subseteq \mathcal{F}_{\Sigma'}$. Then $f = g$ is a conjecture for $\Delta \subseteq \mathcal{F}_{\Sigma'}$.*

PROOF. The result is trivial since the support for the conjecture remains in the extended language, and fact that the database is unchanged ensures that $f = g$ is still not a deductive consequence of $\Delta$. □

We say this is weaker because we cannot extend $\Delta$ in any way since a conjecture can always be falsified (assuming there is more than one entity in its codomain). However, the nature of the language extension itself is less constrained.

An important application of this result is to use it to support language *restriction*. If $\Delta \subseteq \mathcal{F}_\Sigma$ and we can find a $\Sigma'$ such that $\mathcal{F}_{\Sigma'} \subseteq \mathcal{F}_\Sigma$ and $\Delta \subseteq \mathcal{F}_{\Sigma'}$ then we can apply the induction process to the restricted $\Sigma'$ and be assured that any conjectures we obtain will also be conjectures for the original language. We can further use Lemma 5.3 to determine which facts we find will remain facts in $\Sigma$.

This is particularly useful when dealing with infinite domains, such as the natural numbers. If there are an infinite number of entities $a \in \sigma$, for some sort $\sigma$, then following the induction procedure strictly would require an infinite number of function applications to be reduced. This lemma allows us to restrict attention to the finite subset of entities mentioned in $\Delta$.

## 5.3. Applications

Our main interest has been to develop a logical model of reasoning, rather than produce a computationally efficient learning system. However, the performance of IND on more complicated examples has been encouraging. We present the following applications to indicate the potential usefulness of equational induction, and to make some comparisons with other learning methods.

BMWk [**2**] has been used to find recursive relations in term rewriting systems. However, FOIL [**41**], although based on logic programming, has more similarities with our approach and will thus be the main alternative we consider. We believe there are indeed interesting relationships between our work and Inductive Logic Programming (see [**35**] for an overview of ILP), but our prime interest here is in extending our *equational* procedures for deductive reasoning to corresponding procedures for inductive learning. The detailed comparison with ILP will be left to future work.

### 5.3.1. Classification

One of the simplest learning tasks is classification. We illustrate the application of equational induction to classification problems with a machine learning exercise for Prolog [**3**].

It is supposed that a vision system is presented with the image in Figure 5.2, from which it can identify individual objects and determine certain special properties about them. By giving a learning system this information together with a class for each object, we hope that it can associate each class with the corresponding properties which characterize objects within it.

We shall use labels A, ..., L for the objects and represent the information about the objects using the signature

$$\Sigma = \left\{ \begin{array}{l} \mathsf{A}, \ldots, \mathsf{L} \in \mathsf{object}, \\ \mathsf{nut}, \mathsf{key}, \mathsf{screw}, \mathsf{pen}, \mathsf{scissors} : \mathsf{object} \rightarrow \mathsf{sentence}, \\ \mathsf{compact}, \mathsf{long}, \mathsf{otherShaped} : \mathsf{object} \rightarrow \mathsf{sentence}, \\ \mathsf{small}, \mathsf{large} : \mathsf{object} \rightarrow \mathsf{sentence}, \\ \mathsf{noHoles}, \mathsf{oneHole}, \mathsf{twoHoles} : \mathsf{object} \rightarrow \mathsf{sentence} \end{array} \right\}.$$

We provide classification data with rules $\mathsf{nut\ A} \rightarrow \mathsf{True}, \ldots$ and characteristic data by rules $\mathsf{small\ A} \rightarrow \mathsf{True}, \mathsf{noHoles\ A} \rightarrow \mathsf{False}, \ldots$.

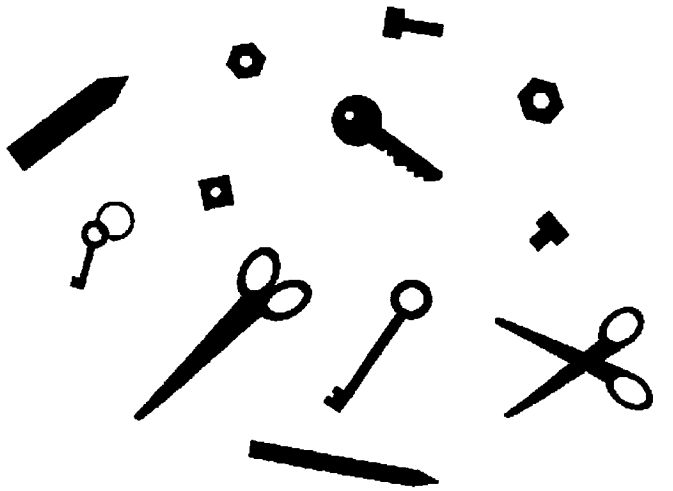


FIGURE 5.2. Visual data for classification (from [3])

From this database, rewrite induction produces 25 equations including the following:

$$\begin{aligned} \mathsf{noHoles} \wedge \mathsf{small} &= \mathsf{screw} \\ \mathsf{twoHoles} \wedge \mathsf{large} &= \mathsf{scissors} \\ \mathsf{noHoles} \wedge \mathsf{large} &= \mathsf{pen} \\ \mathsf{oneHole} \wedge \mathsf{compact} &= \mathsf{nut} \\ \mathsf{otherShaped} &\rightarrow (\mathsf{key} = \mathsf{small}) \\ \mathsf{long} &\rightarrow (\mathsf{key} = \mathsf{oneHole}) \end{aligned}$$

(Here we have written $p \rightarrow (q = r)$ for $p \wedge q = p \wedge r$). The remaining 19 equations express other relations present in the database which are not implied by these classifying equations. Such additional information can often be useful for intermediate proof steps in reasoning. This situation is illustrated in section 5.3.2. At the same time, the generation of the additional information is very expensive, a reflection of the non-guided approach of this induction method, and may well be superfluous.

In comparison, the results given in [3] are

$$\begin{aligned} \mathsf{nut} &\Leftarrow [\mathsf{shape} = \mathsf{compact}, \mathsf{holes} = 1], \\ \mathsf{key} &\Leftarrow [\mathsf{shape} = \mathsf{other}, \mathsf{size} = \mathsf{small}][\mathsf{holes} = 1, \mathsf{shape} = \mathsf{long}], \\ \mathsf{scissors} &\Leftarrow [\mathsf{holes} = 2, \mathsf{size} = \mathsf{large}], \end{aligned}$$

with similar definitions possible for pen and screw. Here the learning is strongly guided by requesting learn(nut), learn(key), etc.

The output generated by FOIL is the following collection of Prolog clauses:

```
nut(A) :- compact(A), onehole(A).
key(A) :- onehole(A), long(A).
key(A) :- other(A), small(A).
screw(A) :- noholes(A), small(A).
pen(A) :- noholes(A), large(A).
scissors(A) :- twohole(A), large(A).
```

FOIL can also be asked to generate clauses for the attributes, though this gives a non-terminating program. Even if this is overcome, the clause definitions are limited in only having one predicate on the left hand side. One of the addition rules generated by rewrite induction is $\mathsf{otherShaped} \rightarrow (\mathsf{scissors} = \mathsf{large})$. Thus if we know an object is other-shaped and large then we can deduce it is a pair of scissors, but also if we know an object is other-shaped and a pair of scissors then we can deduce it is large. This bidirectonal reasoning of equivalence is not possible with the clauses generated by FOIL.

### 5.3.2. Family Relationships

A larger example is one dealing with learning the structure of family trees. This was first given by Hinton [25] as an example of learning using a neural representation, and then by Quinlan [41] as a comparison for FOIL. The aim is not just to learn definitions but to learn definitions from incomplete data and use them to predict the missing information.

The family trees in Figure 5.3 gives information about twelve family relationships: *wife*, *husband*, *mother*, *father*, *daughter*, *son*, *sister*, *brother*, *aunt*, *uncle*, *niece*, and *nephew*. We represent this knowledge as a rewrite database with rules such as $\mathsf{wife\ Marco} \rightarrow \mathsf{Lucia}$. Multivalued functions, such as *aunt* and *uncle*, are expressed using the notion of set, so that $\mathsf{aunt\ Sophia} \rightarrow \{\mathsf{Gina}, \mathsf{Angela}\}$, etc.

In both FOIL and the neural representation of Hinton, it is necessary to give negative information, usually in the form of a closed world assumption. For instance, we would include that the mother of Penelope is not Sophia since it is not specified by the tree. For rewrite induction we only use positive information, assuming that we simply have no knowledge about unspecified function values. (Note that this is different from the implicit entity inequations which only give negative information about entities, not about function values). This can lead to some wild conjectures but such conjectures are often the basis of scientific progress, and indeed in this example we find they help in recovering missing information (while at the same time producing much spurious information).

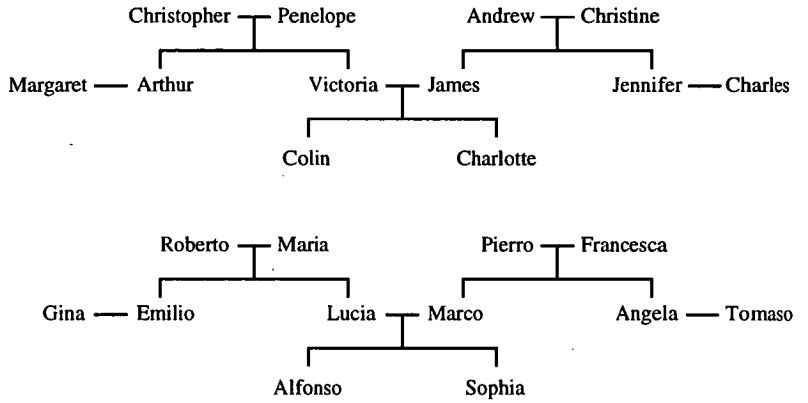



FIGURE 5.3. Two isomorphic family trees (from [**25**])

The two trees in Figure 5.3 specify 104 data rules. Hinton used 100 of these as a training set and then tested to see if the remaining 4 relationships could be found by the trained network. Doing this twice he recorded 7 successes out of 8. Quinlan performed the same experiment twenty times and recorded 78 successes out of 80. Repeating these twenty trials with rewrite induction, all 80 missing relationships were recovered. (This is perhaps lucky. If the two data wife Emilio $\rightarrow$ Gina and husband Gina $\rightarrow$ Emilio, or the other 3 equivalent pairs, are both in the 4 missing then neither can be recovered. The same observation applies to FOIL).

We can repeat this comparison for smaller training sets, where instead of removing just 4 data, we remove 10, 20, 30, ..., 90 of the data. Testing each case 8 times for both FOIL and rewrite induction gave the proportions of recovered data presented in Figure 5.4.

The obvious balance is in the efficiency of the two methods. As a rough estimate of efficiency, running FOIL on a SparcStation 10 to learn rules for the complete family tree takes 3.5 seconds, while the current implementation of IND (in LISP) requires 22.9 seconds.

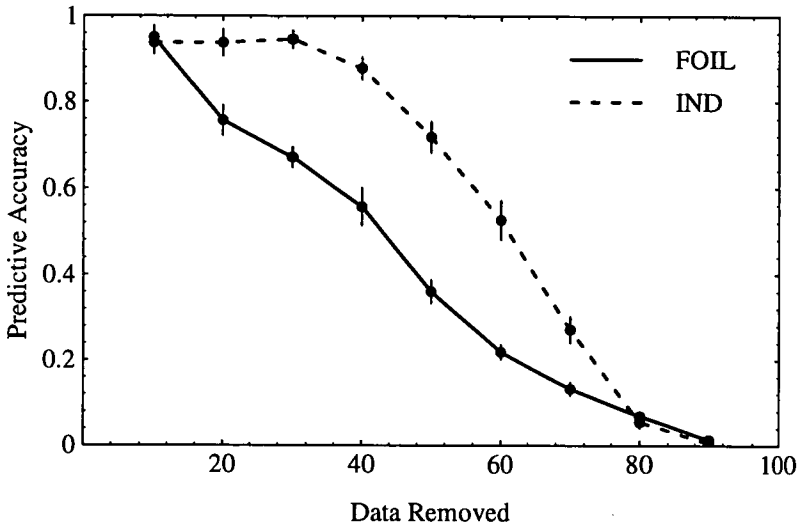



FIGURE 5.4. Predictive performance of FOIL and rewrite induction

CHAPTER 6

# Inductive Belief

In this chapter we turn to various issues that arise from the basic induction algorithm given in Chapter 5. In particular we examine measures of the *belief* we might have in our conjectures. These can be approached both qualitatively and quantitatively. The quality of a system of conjectures can be measured by the kind of predictions it makes, as we will see in Section 6.2. Firstly though we look at measures for the strength of an isolated conjecture.

## 6.1. Numerical Measures of Conjecture Strength

Probability theory has often been used as a tool for analyzing the validity of inductive reasoning, by philosophers from Pascal to Popper. Recently, probabilities have also been used in measuring the quality of rules generated by inductive learning systems, with particular emphasis on Baysian methods [11]. Here we present similar predictive measures, with some additional statistical and information theoretical methods. In each we will see that our typed language allows us to make assumptions which give more precise measures for our models than the standard Baysian approaches.

Each of these measures look at the growing belief in a conjecture as the database $\Delta$ is expanded to include more observations. We are thus interested only in counting the number of *supporting* observations; once an observation is added to $\Delta$ that contradicts the conjecture of interest then that conjecture will no longer be generated by IND.

### 6.1.1. Hypothesis Testing

Suppose we have made a series of $r$ observations which confirm a conjecture $f \simeq g$. If there are $k$ entities in $\mathrm{cod}(f \simeq g)$, $k$ possible values for $f$ and $g$, then the probability that at random the value of each observed $ga$ would match the value of $fa$ is simply $\frac{1}{k}$. Thus the probability that we made the $r$ confirming observations if the values of $f$ and $g$ were random is $(\frac{1}{k})^r$. We call this the *significance value*, $P_r$, for the conjecture based on $r$ observations (with respect to the database $\Delta$).

This definition is motivated by the standard statistical method for testing a conjecture. An untested conjecture has $P_r = 1$, giving certainty that we

could have made the (empty set of) observations we did and thus providing no basis for belief. As this probability decreases, so our belief in the conjecture increases. Figure 6.1 shows the values of $1 - P_r$ for $k = 2$ and $k = 4$. (We have shown $1 - P_r$, where 1 now indicates total belief, for comparison with the following sections). Note that when $k = 4$ it is less likely to make supporting observations by chance than when $k = 2$ and so confidence is gained more quickly.

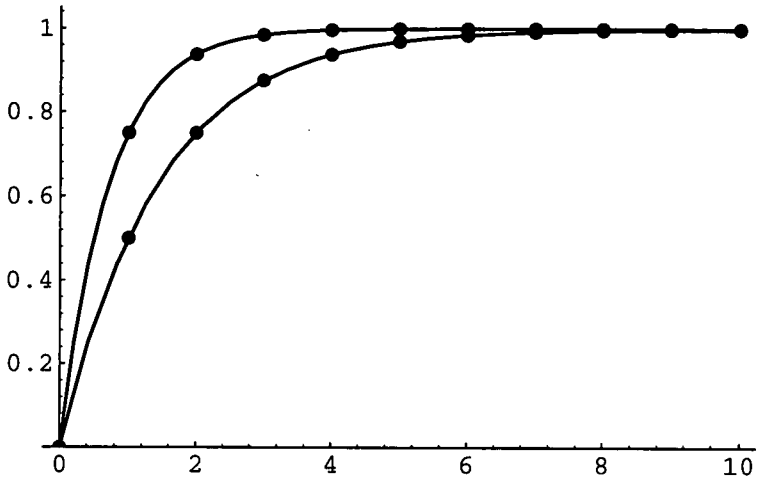



FIGURE 6.1. Progression towards certainty for measure based on hypothesis testing, with $k = 2$ (lower) and $k = 4$ (upper)

Our conjecture male $\simeq$ True ! in Example 5.1 was based on two observations. For each the value of male could have been either True or False and so the probability of making the two supporting observations is $(\frac{1}{2})^2 = .25$. Similarly, the conjecture

fatherinlaw father $\simeq$ father mother

from Example 5.3 was based on a single observation, the application of the equation to Alice. This time though if we knew the value of the left-hand side then there were still six possibilities in $\Sigma$ for the right-hand value, so the probability of the conjecture being made at random was only $\frac{1}{6}$. Thus we would have stronger belief in this second conjecture, based on only one observation, than on the first conjecture which was based on two.

These calculations confirm empirical trials of the induction process. When the database is relatively small there can be a great many 'random' conjectures of predicate relations, most of which disappear as more information is added (as in Examples 5.1 and 5.2). If we observe the value of two functions $f, g : \sigma \rightarrow$ sentence on an entity $a \in \sigma$, then we will make the conjecture $f \simeq g$ with probability $\frac{1}{2}$. That is, for two unrelated sentence-valued functions, we are just as likely to make a conjecture as to not make one.

Conjectures based on larger domains, such as **person**, appear less often and are more likely to remain as the database expands.

The measure $P_r$ is actually a result of the Neyman-Pearson lemma [19]. Let $\theta$ be $P(fa = ga)$, assumed to be common for each entity $a$ in the domain. Then our null hypothesis is that the result of applying $g$ to each entity is independent of the result of applying $f$, reducing to any one of the $k$ entities in $\mathrm{cod}(g)$ with equal probability, i.e. $H_0 : \theta = 1/k$. The alternative hypothesis is that $f \simeq g$ is true, holding for all entities in $\mathrm{dom}(f)$, so we have $H_1 : \theta = 1$. For these simple hypotheses the lemma gives $(1/k)^r$ as the best test statistic to use to decide between them.

Suppose that there are $n$ entities in $\mathrm{dom}(f \simeq g)$. Then the probability $P_r$ is still meaningful when $r = n$ but assumes that the $n$ entities are only a sample from a larger universe, such as the type **person**. If the $n$ entities are in fact the complete presentation of a finite domain, as in **sentence**, then we would somehow like to express this completeness in the value of $P_r$. The approach in the next section addresses this point.

Finally note that we are assuming here that both $\mathrm{dom}(f)$ and $\mathrm{cod}(f)$ are finite. This is justified by Lemma 5.3 as our use of a finite database allows the restriction of language to the entities mentioned in the database. We will typically use this restricted size of $\mathrm{cod}(f)$ since it gives the observed size of the sort. (See Section 6.4.4 for a detailed example using an infinite domain.)

### 6.1.2. Posterior Measures

Another way to view the establishment of belief in a conjecture is as finding the probability of the conjecture being true *conditional on* the observations made. Such conditional probabilities can be calculated using the discrete form of Bayes's rule

$$P(B_0|A) = \frac{P(A|B_0)P(B_0)}{\sum_j P(A|B_j)P(B_j)}.$$

The value we will determine is the probability that there are no falsifying cases in the $n$ entities (event $B_0$) given that we have observed no falsifying cases in a sample of $r$ entities (event $A$). To calculate the conditional probabilities $P(A|B_j)$, the probabilities of observing no falsifying cases if there are $j$ falsifying cases present, we use the model that each entity is either a supporting case or a falsifying case, and that $\mathrm{dom}(f \simeq g)$ gives a set of such cases. Sampling the entities is then equivalent to successively taking elements from the set, without replacement, and seeing whether they support or falsify $f \simeq g$. The values of $P(A|B_j)$ are then given by the

hypergeometric distribution

$$P(A|B_j) = \binom{n-r}{j} \Big/ \binom{n}{j}.$$

The remaining terms in Bayes's rule are the prior distribution probabilities $P(B_j)$. For these we must make an assumption about the distribution of the number of supporting/falsifying cases present without reference to the conjecture.

A uniform prior distribution has it equally likely to find the world having any number $j$ of falsifying examples, $j = 0, \ldots, n$, giving $P(B_j) = 1/(n+1)$. The corresponding posterior probability of there being no falsifying examples having observed $r$ supporting cases is then

$$P(B_0|A) = \frac{r+1}{n+1}.$$

This gives a linear progression towards certainty, with unity obtained if all $n$ cases are found to be supporting. Figure 6.2 illustrates the growing confidence in a conjecture when $n = 10$.

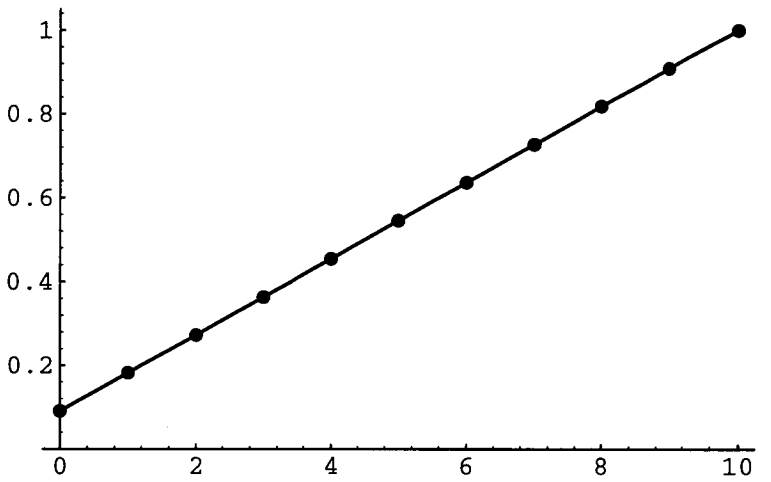



FIGURE 6.2. Progression towards certainty for posterior measure based on a uniform prior distribution ($n = 10$)

An alternative is to view each case as being equally likely to be supporting or falsifying (or perhaps supporting with probability $1/k$). Then the prior distribution is binomial, with $P(B_j) = \binom{n}{j}(1/2)^n$, giving posterior probability

$$P(B_0|A) = 2^{r-n}.$$

Again we have a progression to unity, this time along an exponential curve. It is most likely that half of the cases would confirm the conjecture, so we find that until we have examined half of the cases the certainty increases

very slowly. Passing that point it then increases steeply to 1, as shown in Figure 6.3.

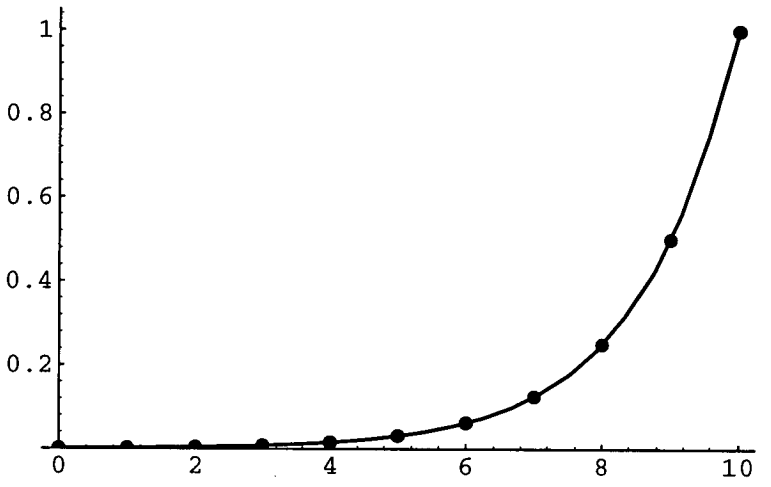



FIGURE 6.3. Progression towards certainty for posterior measure based on a binomial prior distribution ($n = 10$)

### 6.1.3. Predictive Measures

The way a reasoner should behave is perhaps the inverse of that given by the binomial posterior measure above. Instead of waiting until it is safe to start establishing certainty, typically a few supporting observations are sufficient to give a strong feeling of truth, with subsequent observations being less and less 'surprising' and thus containing less information about the truth of the conjecture.

We can obtain a measure with these properties using an urn model similar to those in [**18**]. Suppose a red ball represents a supporting case and a white ball represents a falsifying case. If there are $n$ cases in all then we view the world as an urn containing $n$ balls, each either red or white. We don't known how many red balls there are and so we sample the urn by examining the cases it contains (without replacement). A true conjecture (a fact) corresponds to the urn containing only supporting cases (red balls).

For two colours there are $n + 1$ possible urns labelled $j = 0, \ldots, n$, with the $j$th urn containing $j$ supporting cases and $n - j$ non-supporting cases. Again it is straightforward to use Bayes's rule to find a measure of truth. This time successive observations reduce the possibilities for which urn we are sampling from and so increase the probability of the next case being a supporting one. The worth of a conjecture is its ability to predict, and so we take as our belief measure the probability $P(A_{r+1}|A)$ that the next observed case is supporting given that we have observed $r$ supporting cases.

The probability of $A_r$, finding $r$ supporting cases, under the assumption that each urn is equally probably is

$$P(A_r) = \frac{1}{n+1} \sum_{j=r}^{n} \frac{j(j-1)\cdots(j-r+1)}{n(n-1)\cdots(n-r+1)}.$$

(The probability of finding $r$ red balls in an urn with $j < r$ is 0). If the next case holds then we will have observed $r+1$ supporting cases and so the required probability is $P(A_{r+1}|A_r) = P(A_{r+1})/P(A_r)$. Thus

$$P(A_{r+1}|A_r) = \frac{r+1}{r+2}.$$

As the number of observations increases, so the rate of increase of belief decreases, as illustrated in Figure 6.4. This matches our notion of 'information content' outline above, the later observations contributing less to our belief. Note that when $r = 0$, when no observations have been made, we have $P(A_1|A) = \frac{1}{2}$. Thus we can only say that the next observed case will be supporting or falsifying with equal probability, so that our conjecture contains no predictive information. This is one reason why our definition of conjecture (Definition 5.2) includes the requirement that there is at least one supporting observation.

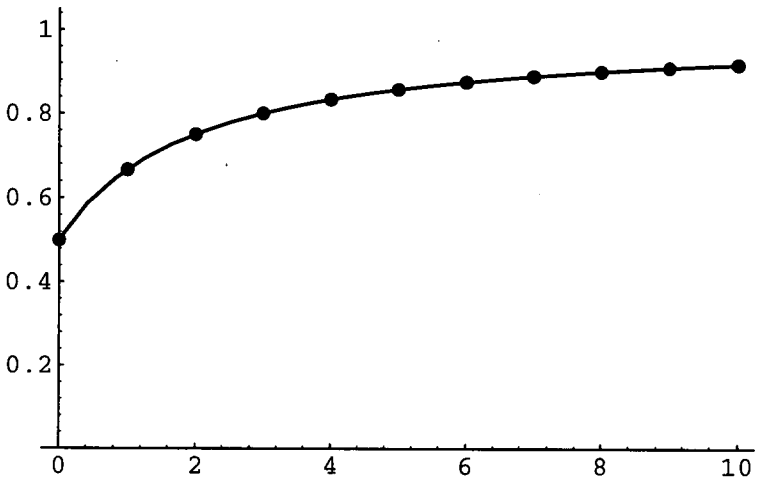



FIGURE 6.4. Progression towards certainty for measure based on predictive power

We can make this measure more accurate by taking into account the size of the conjecture's codomain, as we did in the hypothesis testing of Section 6.1.1. That is, we would like to incorporate the fact that a conjecture such as **male father** $\simeq$ **True** ! is more likely to hold at random than a conjecture **wife father** $\simeq$ **mother**. To do this we extend our urn model to balls with $k$ possible colours, where $k$ is the number of entities in the codomain of the conjecture. One of the $k$ colours represents a supporting case, the rest

being falsifying cases. If we have $n$ cases in all then $N_n^k$, the number of different urns with $n$ balls coloured by $k$ colours is given the by $n$-dimensional tetrahedral numbers (see [1])

$$N_n^k = \binom{k+n-1}{n}.$$

Thus $N_n^k$ is the total number of possible states of the world, all of which we assume are equiprobable. The number of these states with $j$ supporting cases is given by $N_{n-j}^{k-1}$, so that we now have

$$P(A_r) = \frac{1}{N_n^k} \sum_{j=r}^{n} N_{n-j}^{k-1} \frac{j(j-1)\cdots(j-r+1)}{n(n-1)\cdots(n-r+1)}.$$

Using Bayes's rule as before gives

$$P(A_{r+1}|A_r) = \frac{r+1}{r+k}.$$

When $k = 2$ we have the same result as before, so this new formula can be seen as a generalization of the predictive measure from predicates to arbitrary function equations.

Figure 6.5 illustrates the growth of certainty for conjectures with $k = 3$ and $k = 4$. Since such conjectures are less likely to hold at random than predicates (where $k = 2$) there is a corresponding decrease in their predictive strength. When $r = 0$ we have $P(A_1|A_0) = \frac{1}{k}$ so as before an untested conjecture has no predictive information.

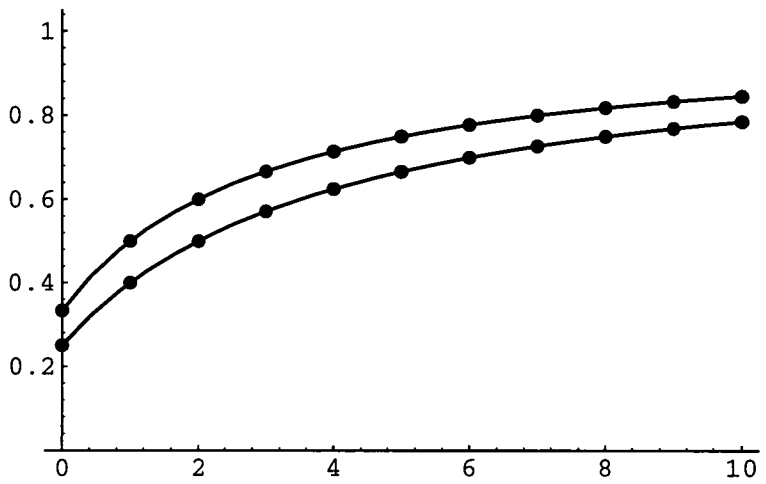



FIGURE 6.5. Progression towards certainty for general predictive measure

### 6.1.4. Information Content

In the previous section we described supporting observations that came later as being less 'surprising' than earlier ones. This was the motivation for our predictive measure, but surprise itself is also a central part of *information theory* [**46**]. We can use our predictive probability $P(A_{r+1}|A_r)$ to calculate the information content of the next case being a supporting case and also the entropy of the next observation as a whole.

Suppose we have made $r$ supporting observations for a conjecture with $k$ entities in its codomain. The *self-information* [**22**] of the event that the next observation is supporting is given by

$$I_r^k = -\log_2 P(A_{r+1}|A_r) = -\log_2 \left(\frac{r+1}{r+k}\right).$$

Figure 6.6 shows values of $I_r^k$ for increasing $r$ when $k = 2, 3, 4$. As the number of observations grows the amount of information provided by each new supporting observation decreases. That is, the fact that the new observation supports the conjecture becomes less surprising, corresponding to the increasing predictive strength of the conjecture. Note too that for larger values of $k$ the outcome is each observation is always more surprising, since there are more possible outcomes, and hence the self-information values are above those for smaller $k$.

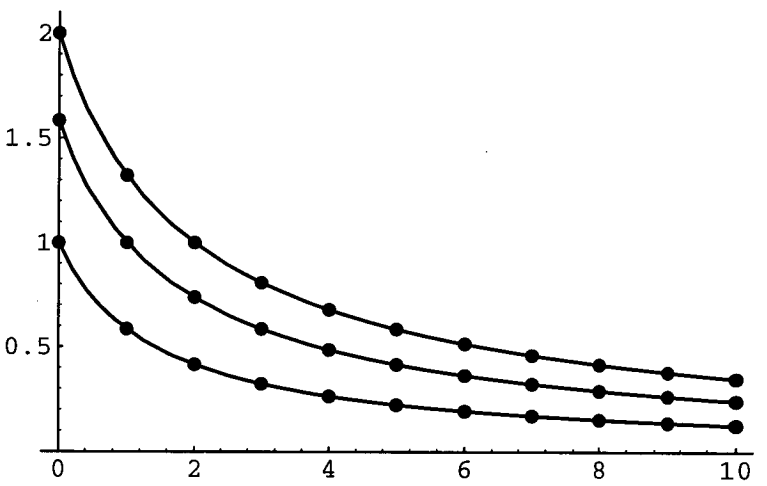



FIGURE 6.6. Declining self-information of supporting observations as $r$ increases, showing $k = 2$ (lower), $k = 3$, and $k = 4$

Self-information gives the amount of information contained in a *supporting* observation, but what can we say about the observation in general? A standard way of measuring the *certainty* we have in the outcome of the next observation is to look at the *entropy* of the observation. When entropy,

the expected self-information for all outcomes, is 0 we have total certainty about the result. For larger entropies our corresponding certainty is less. If we suppose that each of the $k-1$ non-supporting outcomes is equally likely then after $r$ supporting observations the entropy of the next observation is

$$H_r^k = -(P \log_2 P) - (1-P) \log_2 \frac{1-P}{k-1},$$

where $P = P(A_{r+1}|A_r)$ is the probability that the next observation is supporting.

Figure 6.7 shows the decreasing entropy for $k = 2, 3, 4$. As before larger values of $k$ imply greater uncertainty and so give higher entropy values. Note also that for an untested conjecture we have no predictive information about the next observation and so the self-information and entropy values coincide.

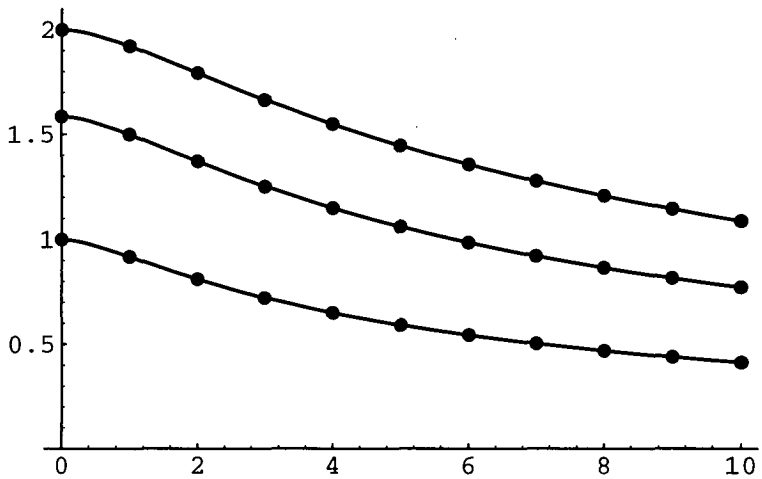



FIGURE 6.7. Declining entropy of observations as $r$ increases, showing $k = 2$ (lower), $k = 3$, and $k = 4$

To conclude this section it is worth noting that in practice we will not make any use of these measures. Conjectures are typically not made in isolation, but rather as part of a conjectured *theory* of function equations. However, the ideas here will form the basis of our discussion of such theories, particularly the null assumption of a function's result being randomly drawn from the entities of its codomain. Furthermore, the graphs in the preceding figures give a good illustration of the growth of belief in conjectured knowledge. For varying $k$ we see the general number of observations we must make to achieve a given certainty. In statistical terms, this gives the *sample size* needed for a given $k$ to achieve such certainty.

## 6.2. Consistency and Experiments

### 6.2.1. Closure

We define the *closure*, $C_E(\Delta)$, of a database $\Delta$ with respect to a set of function equations $E$ to be the set of all data that can deduced from $E$ and $\Delta$. This operation is similar to the Knuth-Bendix completion procedure except that we look for *all* grounded consequences, rather than just a reduced set.

In forming the closure we implicitly have the entity inequations $a \neq b$ for all $\Sigma$-entities $a, b$, with $a \not\equiv b$, in the same sort $\sigma$. The closure process may then generate data which contradict the original data, as happened in the database $\Delta$ in Example 5.2. We then say the closure *failed* and write $C_E(\Delta) = \perp$. The corresponding $E$ is said to be *inconsistent* with respect to the database $\Delta$. If $C_E(\Delta) \neq \perp$ then we say $E$ is *consistent* with respect to $\Delta$.

Note that $C_E(\Delta)$ need not be finite. If

$$\Delta = \{\mathsf{not\ True} \to \mathsf{False},\ \mathsf{not\ False} \to \mathsf{True}\}$$

and $E = \phi$ then, for example, the datum $\mathsf{not}^{2k}\ \mathsf{True} \to \mathsf{True}$ can be deduced from $E \cup \Delta$ for any $k \geq 1$. Thus in defining the closure process we fix attention on data with a fixed function length, as we did for the induction process itself.

Let $R = \mathrm{KBI}(E \cup \Delta)$. If $R$ is $\perp$ then $E$ is inconsistent with $\Delta$ and the closure failed. Otherwise the closure of $\Delta$ of length $m$ with respect to $E$ is given by

$$C_E^m(\Delta) = \{fa = b \mid \delta(f) = m, a \in \mathrm{dom}(f), b \in \mathrm{cod}(f), fa \to_R b\}.$$

If $\Sigma$ is finite then we immediately have that $C_E^m(\Delta)$ is finite since there are only finitely many functions $f$ of length $m$. Thus for fixed $m$ the closure process will terminate. However, as noted above, we may have $C_E^M(\Delta) \neq \phi$ for any $M$.

Our main interest will be in the case when $m = 1$. Then the closure gives all known and predicted values for processes represented by single functions in $\Sigma$. We will use $C_E(\Delta)$ for $C_E^1(\Delta)$.

The following two lemmas are trivial consequences of the above definition but give important properties of the closure.

LEMMA 6.1. *If $C$ is consistent with respect to $\Delta$ then*

$$\Delta \subseteq C_C(\Delta).$$

PROOF. By the monotonicity of $\Rightarrow$ (Theorem 4.1), any consequence of $\Delta$ is a consequence of $C \cup \Delta$. □

LEMMA 6.2 (Hereditary). *If $C$ is consistent with respect to $\Delta$ and $C' \subseteq C$ then $C'$ is also consistent with respect to $\Delta$*

PROOF. Anything generated by KBI for $C'$ is a subset of that created for $C$ and so if $C$ did not activate *Contradiction* then neither will $C'$. □

The hereditary property of Lemma 6.2 gives some indication that the consistent sets for a database $\Delta$ may be the independent sets of a matroid [**36**]. However, the matroid exchange property is not satisfied, as shown in the following:

EXAMPLE 6.1 (Consistent subsets). Consider again the database $\Delta$ in Example 5.2. We found the five conjectures

$$C = \left\{ \begin{array}{l} \mathsf{female} \simeq \mathsf{False}\ !,\ \mathsf{not\ female} \simeq \mathsf{True}\ !, \\ \mathsf{female} \simeq \mathsf{not\ male},\ \mathsf{not\ female} \simeq \mathsf{male}, \\ \mathsf{not\ not} = \mathsf{i} \end{array} \right\}.$$

The subsets of $C$ which are consistent with respect to $\Delta$ are

$$\begin{array}{l} \phi, \{\mathsf{not\ not} = \mathsf{i}\}, \\ \{\mathsf{female} \simeq \mathsf{False}\ !\}, \{\mathsf{not\ female} \simeq \mathsf{True}\ !\}, \\ \{\mathsf{female} \simeq \mathsf{not\ male}\}, \{\mathsf{not\ female} \simeq \mathsf{male}\}, \\ \{\mathsf{not\ not} = \mathsf{i}, \mathsf{female} \simeq \mathsf{False}\ !\}, \\ \{\mathsf{not\ not} = \mathsf{i}, \mathsf{not\ female} \simeq \mathsf{True}\ !\}, \\ \{\mathsf{female} \simeq \mathsf{False}\ !, \mathsf{not\ female} \simeq \mathsf{True}\ !\}, \\ \{\mathsf{not\ not} = \mathsf{i}, \mathsf{female} \simeq \mathsf{not\ male}\}, \\ \{\mathsf{not\ not} = \mathsf{i}, \mathsf{not\ female} \simeq \mathsf{male}\}, \\ \{\mathsf{female} \simeq \mathsf{not\ male}, \mathsf{not\ female} \simeq \mathsf{male}\}, \\ \{\mathsf{not\ not} = \mathsf{i}, \mathsf{female} \simeq \mathsf{False}\ !, \mathsf{not\ female} \simeq \mathsf{True}\ !\}, \\ \{\mathsf{not\ not} = \mathsf{i}, \mathsf{female} \simeq \mathsf{not\ male}, \mathsf{not\ female} \simeq \mathsf{male}\}. \end{array}$$

A collection of sets $\mathcal{I}$ satisfy the *exchange property* if for any $I, J \in \mathcal{I}$ with $|I| = |J| + 1$ we can choose an $e \in I \setminus J$ such that $J \cup \{e\}$ is in $\mathcal{I}$. For the above collection of sets consider

$$\begin{array}{l} I = \{\mathsf{not\ not} = \mathsf{i}, \mathsf{female} \simeq \mathsf{False}\ !, \mathsf{not\ female} \simeq \mathsf{True}\ !\}, \\ J = \{\mathsf{not\ not} = \mathsf{i}, \mathsf{not\ female} \simeq \mathsf{male}\}. \end{array}$$

Neither $\mathsf{female} \simeq \mathsf{False}\ !$ nor $\mathsf{not\ female} \simeq \mathsf{True}\ !$ is consistent with $J$ and so the collection does not satisfy the exchange property. ◇

Lemma 6.2 has an important application in that, because they are hereditary, we can express the above list of all consistent subsets by giving just the maximally consistent subsets. This will be our topic in the next section. The fact that the subsets don't constitute a matroid means that there is no simple way of finding these maximal subsets, by using a greedy algorithm [**36**], and so we will need some combinatorial approach.

Note that a set $E$ of conjectures and facts is necessarily consistent since it contains only function equations and so cannot produce any ground equations to contradict declared entity inequations. Thus $\mathrm{KBI}(E)$ is never $\perp$ and so we can use $\mathrm{KB}(E)$ as a canonical form of our conjectures.

LEMMA 6.3. *A set $C$ is consistent with respect to $\Delta$ if and only if $KB(C)$ is consistent with respect to $\Delta$.*

PROOF. Since $\Rightarrow$ is a consequence relation, $C \cup \Delta$ and $\mathrm{KB}(C) \cup \Delta$ are equivalent. □

Reducing a set of conjectures is not always desirable, especially in the presence of erasing conjectures such as **male** $\simeq$ **True** !. In particular it excludes the selection of working theories from the conjectures since one theory may destroy another theory in the process. (This is the case with the conjectures of Example 5.2.) The conjectures should only be reduced if they are consistent with the data and hence constitute a theory in themselves.

A final definition to consider in relation to closure is the motivation for the whole exercise.

DEFINITION 6.1. If $E$ is consistent with respect to $\Delta$ then we define the *predictions*, $\mathcal{P}_E(\Delta)$, of $E$ with respect to $\Delta$ as

$$\mathcal{P}_E(\Delta) = \mathcal{C}_E(\Delta) \setminus \mathcal{D}_\Delta.$$

That is, the predictions are the data implied by $E$ and $\Delta$ which are not already known. Predictions will play an important role in describing experiments and providing a qualitative measure of the strength of a system of conjectures.

Until now we have allowed our database $\Delta$ to be quite general in the type of equations it contains. However, much of our interest lies in databases which model the making of observations about particular processes, where a process is represented by a single function. Each datum will thus involve only a single function, such as the observation **father Alice** $\rightarrow$ **John** about the process **father**. We capture this kind of database with the following definition.

DEFINITION 6.2 (Simple Database). If every datum $fa \rightarrow b$ in a database $\Delta$ has a $\Sigma$-word $f$ then we say that $\Delta$ is *simple*.

Simple databases also give rise to useful results, such as the following characterization of facts in terms of their predictions.

LEMMA 6.4 (Summative Induction). *If $f = g$ is a fact from a simple database $\Delta$ then $\mathcal{P}_{(f=g)}(\Delta) = \phi$.*

PROOF. Suppose $ha = b$ is in $\mathcal{C}_{(f=g)}(\Delta)$, so $\Delta \cup (f = g) \Rightarrow ha = b$, and that $\Delta \not\Rightarrow (ha = b)$. Thus a proof of $ha = b$ requires the use of $f = g$ in at least one proof step. That is, for some $h_1$, $h_2$, and $c$,

$$ha =_{\Delta \cup (f=g)} h_1 f h_2 c = h_1 g h_2 c =_{\Delta \cup (f=g)} = b.$$

To reduce $h_1 g h_2 c$ to $b$ we must know the value of $h_2 c$, since $\Delta$ is simple. Suppose $h_2 c = c'$. Then $h_1 f c' =_\Delta h_1 g c'$, since $f = g$ is a fact from $\Delta$, and so the use of $f = g$ was unnecessary in this proof step. This holds for all uses of $f = g$ and so $ha =_\Delta b$, contradicting our assumption. Hence there is nothing in the closure of $\Delta$ with $f = g$ that is not a consequence of $\Delta$, giving $\mathcal{P}_{(f=g)}(\Delta) = \phi$. □

To see that the simplicity requirement is necessary, consider the database

$$\Delta = \left\{ \begin{array}{l} \mathsf{male\ John} \rightarrow \mathsf{True},\ \mathsf{tall\ John} \rightarrow \mathsf{True}, \\ \mathsf{male\ father\ John} \rightarrow \mathsf{True},\ \mathsf{not\ tall\ father\ John} \rightarrow \mathsf{False} \end{array} \right\},$$

from which we induce the single fact $\mathsf{male} = \mathsf{tall}$. We can then use this fact to predict $\mathsf{not\ True} = \mathsf{False}$ using the the two non-simple data in $\Delta$.

This lemma justifies referring to facts as a form of *summative*, rather than ampliative, induction. This may be somewhat unfair however, since a fact may arise simply because the underlying signature has not been completely defined. As seen in Lemma 5.3, a language extension may downgrade a fact to a conjecture, possibly giving predictions for the newly added entities.

### 6.2.2. Competing Theories

DEFINITION 6.3. A subset $C' \subseteq C$ is *maximally consistent* in $C$ with respect to $\Delta$ if $C'$ is consistent with respect to $\Delta$ and there is no $e \in C \setminus C'$ such that $C' \cup e$ is consistent with respect to $\Delta$.

When the $\Delta$ is clear we will say simply that $C'$ is a maximally consistent subset of $C$.

LEMMA 6.5. *Suppose $C$ is the result of induction on $\Delta$. Then if $C$ is inconsistent with respect to $\Delta$, $C$ contains at least two maximally consistent subsets.*

PROOF. Any single conjecture $c_1 \in C$ must be consistent with respect to $\Delta$, since it was the result of induction on $\Delta$. Hence there must be a maximally consistent $C_1 \subset C$ containing $c_1$, obtained by successively adding conjectures to $\{c_1\}$ from $C$ until no longer consistently possible. Since $C$ is not consistent there exists another $c_2 \in C \setminus C_1$, and similarly there must be a maximally consistent $C_2 \subset C$ containing $c_2$. Since $c_2 \notin C_1$, $C_1$ and $C_2$ are two distinct maximally consistent subsets of $C$. □

Hence if a set of induced beliefs are inconsistent, there are at least two smaller belief sets which are consistent. We can view these smaller sets as *competing theories* for explaining the data.

DEFINITION 6.4. The *theories*, $\mathcal{T}_C(\Delta)$, of a set of conjectures $C$ with respect to a database $\Delta$, is the set of maximally consistent subsets of $C$ with respect to $\Delta$. The *theories* of a database $\Delta$ is the set $\mathcal{T}(\Delta) = \mathcal{T}_{\mathrm{IND}(\Delta)}(\Delta)$.

Hence a set of conjectures $C$ is consistent with respect to $\Delta$ if and only if $|\mathcal{T}_C(\Delta)| = 1$.

EXAMPLE 6.2 (Competing theories). Consider again the database $\Delta$ in Example 5.2. We found the five conjectures

$$C = \left\{ \begin{array}{l} \mathsf{female} \simeq \mathsf{False}\ !,\ \mathsf{not\ female} \simeq \mathsf{True}\ !, \\ \mathsf{female} \simeq \mathsf{not\ male},\ \mathsf{not\ female} \simeq \mathsf{male}, \\ \mathsf{not\ not} = \mathsf{i} \end{array} \right\}.$$

Obtaining maximally consistent subsets gives

$$\mathcal{T}_C(\Delta) = \left\{ \begin{array}{l} \{\mathsf{not\ not} = \mathsf{i}, \mathsf{female} \simeq \mathsf{False}\ !, \mathsf{not\ female} \simeq \mathsf{True}\ !\}, \\ \{\mathsf{not\ not} = \mathsf{i}, \mathsf{female} \simeq \mathsf{not\ male}, \mathsf{not\ female} \simeq \mathsf{male}\} \end{array} \right\}.$$

The two subsets correspond to two differing theories about the function female. One theory says that every person is female, while the other says that to be female is to not be male. A reasoner must chose one of these as a *working theory*. $\diamond$

EXAMPLE 6.3 (Three competing theories). The example we have used so far looks at functions with results in sentence, either True or False, but inconsistency can obviously arise for arbitrary functions. Consider the larger database

$$\Delta = \left\{ \begin{array}{l} \mathsf{mother\ Paul} \rightarrow \mathsf{Mary}, \\ \mathsf{grandfather\ Paul} \rightarrow \mathsf{Bob},\ \mathsf{daughter\ Bob} \rightarrow \mathsf{Mary}, \\ \mathsf{wife\ Peter} \rightarrow \mathsf{Mary},\ \mathsf{father\ Paul} \rightarrow \mathsf{Peter}, \\ \mathsf{aunt\ Paul} \rightarrow \mathsf{Jill},\ \mathsf{sister\ Jill} \rightarrow \mathsf{Mary}, \\ \mathsf{grandfather\ Alice} \rightarrow \mathsf{George},\ \mathsf{daughter\ George} \rightarrow \mathsf{Jenny}, \\ \mathsf{father\ Alice} \rightarrow \mathsf{John},\ \mathsf{wife\ John} \rightarrow \mathsf{Jill}, \\ \mathsf{aunt\ Alice} \rightarrow \mathsf{Mary},\ \mathsf{sister\ Mary} \rightarrow \mathsf{Kelly} \end{array} \right\}.$$

The first seven data use incomplete information to teach the reasoner three different ways of finding a person's mother. The last six then play on the incompleteness to give inconsistency in the predictions of Alice's mother. We have

$$C = \mathrm{IND}(\Delta, 2) = \left\{ \begin{array}{l} \mathsf{daugher\ grandfather} \simeq \mathsf{mother}, \\ \mathsf{wife\ father} \simeq \mathsf{mother}, \\ \mathsf{sister\ aunt} \simeq \mathsf{mother} \end{array} \right\}$$

giving three competing theories

$$\mathcal{T}_C(\Delta) = \left\{ \begin{array}{l} \{\mathsf{daugher\ grandfather} \simeq \mathsf{mother}\}, \\ \{\mathsf{wife\ father} \simeq \mathsf{mother}\}, \\ \{\mathsf{sister\ aunt} \simeq \mathsf{mother}\} \end{array} \right\}.$$

◇

As mentioned previously, without noise the only cause of inconsistency in an inductive belief set is incomplete information. We thus take the scientific approach and seek to make further observations to remove the inconsistency and so establish one of the competing theories as the only consistent theory. This is our subject in the next section.

### 6.2.3. Inductive Experiments

In the case of deductive reasoning, we saw in Section 3.5 that when a possible theorem was a non-theorem we were able to generate certain *side conditions* which could be added to the hypotheses to give a theorem. This is a meta-reasoning for deduction whereby the reasoner is not only able to prove theorems but is also able to say what is wrong when it fails to prove a theorem.

The analogous meta-reasoning for induction is the generation of *experiments*. Given a database, the reasoner attempts to induce a belief set from the information it contains. If the resulting set of conjectures is consistent with respect to the database then we would say the reasoner has been *successful* in establishing a set of beliefs. However if inconsistency is present in the conjectured set then the result of the inductive process is unsuccessful, with belief to be given to one of possibly many maximally consistent subsets.

In this case the reasoner needs to be able to decide between the potential belief sets, and the scientific approach to resolving inconsistent theories is experimentation. Like the generation of side conditions, the reasoner needs to generate experiments that, when carried out, it thinks will result in a single consistent set of conjectured belief.

DEFINITION 6.5. An *experiment* is a composition of a function term $f : \sigma \rightarrow \tau$ and an entity $a \in \sigma$. The *result* of an experiment $fa$ is an equation $fa = b$ for some entity $b \in \tau$.

We are interested firstly in choosing an experiment to perform and then being able to make some comments about the change that adding the result to the database will have on the generated theories. Of course these two questions are closely related since ideally we want to choose experiments that reduce the number of competing theories.

DEFINITION 6.6 (Crucial experiments). Suppose $\mathcal{T}(\Delta) = \{C_1, \ldots, C_n\}$. If there is some experiment $fa$ such that $(fa = b) \in \mathcal{P}_{C_j}(\Delta)$ and $(fa = c) \in \mathcal{P}_{C_k}(\Delta)$ for some $b \not\equiv c$, then $fa$ is a *crucial* experiment for $\Delta$.

The motivation for this definition is obvious. If $C_j \cup \Delta \Rightarrow (fa = b)$ and $C_k \cup \Delta \Rightarrow (fa = c)$ then the result of $fa$ must contradict $C_j$ or $C_k$, or possibly both. The intrinsic belief dynamics of applying IND to the new database will then revise $C_j$ and $C_k$, finding a maximal subset of each which is consistent with the result of $fa$. Both will decrease in size and ultimately one or both will disappear.

EXAMPLE 6.4 (Conflicting predictions). Consider again the database

$$\Delta = \left\{ \begin{array}{l} \mathsf{male\ Paul} \rightarrow \mathsf{True},\ \mathsf{male\ Alice} \rightarrow \mathsf{False}, \\ \mathsf{female\ Paul} \rightarrow \mathsf{False}, \\ \mathsf{not\ False} \rightarrow \mathsf{True},\ \mathsf{not\ True} \rightarrow \mathsf{False} \end{array} \right\}.$$

We have two maximally consistent beliefs sets

$$C_1 = \{\mathsf{female} \simeq \mathsf{False}\ !, \mathsf{not\ female} \simeq \mathsf{True}\ !, \mathsf{not\ not} = \mathsf{i}\}, \text{ and}$$
$$C_2 = \{\mathsf{female} \simeq \mathsf{not\ male}, \mathsf{not\ female} \simeq \mathsf{male}, \mathsf{not\ not} = \mathsf{i}\}.$$

Generating the closures of both sets with respect to $\Delta$ we find that

$$\mathcal{P}_{C_1}(\Delta) = \{\mathsf{female\ Alice} \rightarrow \mathsf{False}\}, \text{ and}$$
$$\mathcal{P}_{C_2}(\Delta) = \{\mathsf{female\ Alice} \rightarrow \mathsf{True}\}.$$

Hence have the single crucial experiment **female Alice**. Following the experiment we will have a single consistent theory, either $C_1$ or $C_2$ for Alice being either male or female, respectively. $\diamond$

EXAMPLE 6.5 (Three conflicting predictions). For the three competing theories given in Example 6.3, we find three different predictions for the mother of Alice:

$$\mathsf{mother\ Alice} \rightarrow \mathsf{Jenny},\ \mathsf{mother\ Alice} \rightarrow \mathsf{Jill},\ \mathsf{mother\ Alice} \rightarrow \mathsf{Kelly}.$$

The crucial experiment to consider is **mother Alice**. $\diamond$

The situation is not always as straightforward as in these examples. The presence of competing theories does not always imply the presence of crucial experiments, as we will see later in some extended examples, and so we cannot guarantee the reduction of the theories. However, it is still valuable to look at the predictions of the theories for a source of experiments. Each prediction is essentially an opportunity for falsification, a notion we will return to in Section 6.2.4. We make a null hypothesis that the result of $fa$ is equally likely to be any of the $b \in \mathrm{cod}(f)$, all but one of which will falsify the theory. Thus it is better to try experiments with large codomains first (such as **person**) rather than those with small codomains (such as the boolean **sentence**).

One special class of non-crucial experiments is worth considering. In Section 3.5 we identified certain side conditions as being *extraneous*, specifically those which were consequences of the hypotheses alone and thus would give a theorem for any conclusion. An analogous notion exists for inductive reasoning.

DEFINITION 6.7 (Extraneous Result). Suppose that for a database $\Delta$ we have the theories

$$\mathcal{T}(\Delta) = \{C_1, C_2, \ldots, C_n\}.$$

If $(fa = b) \in \mathcal{P}_{C_k}(\Delta)$ for some $k$ and there is no $j$ and $c \not\equiv b$ such that $(fa = c) \in \mathcal{P}_{C_j}(\Delta)$ then the experimental result $fa = b$ is said to be *extraneous*.

EXAMPLE 6.6 (Extraneous result). We introduce a new person, George, to the discussion of Example 6.4, adding to $\Delta$ that George is male to give

$$\Delta' = \left\{ \begin{array}{l} \mathsf{male\ Paul} \rightarrow \mathsf{True},\ \mathsf{male\ Alice} \rightarrow \mathsf{False}, \\ \mathsf{male\ George} \rightarrow \mathsf{True},\ \mathsf{female\ Paul} \rightarrow \mathsf{False}, \\ \mathsf{not\ False} \rightarrow \mathsf{True},\ \mathsf{not\ True} \rightarrow \mathsf{False} \end{array} \right\}.$$

We have the same consistent sets $C_1$ and $C_2$ as before but this time we find

$$\mathcal{P}_{C_1}(\Delta') = \{\mathsf{female\ Alice} \rightarrow \mathsf{False},\ \mathsf{female\ George} \rightarrow \mathsf{False}\},\ \text{and}$$
$$\mathcal{P}_{C_2}(\Delta') = \{\mathsf{female\ Alice} \rightarrow \mathsf{True},\ \mathsf{female\ George} \rightarrow \mathsf{False}\},$$

so the experimental result $\mathsf{female\ George} = \mathsf{False}$ is extraneous. $\diamond$

The observation of an extraneous result will not distinguish between theories but may still result in new conjectures.

We cannot talk about extraneous *experiments* since if all competing theories make the same prediction for the result of an experiment and, instead of being extraneous, the result *differs* from the common prediction, then the result has a drastic effect on the inductive beliefs.

DEFINITION 6.8 (Anomalous Result). Suppose that for a database $\Delta$ we have the theories

$$\mathcal{T}(\Delta) = \{C_1, C_2, \ldots, C_n\}.$$

If $(fa = b) \in \mathcal{P}_{C_k}(\Delta)$ for some $k$ and there is no $j$ and $c \not\equiv b$ such that $(fa = c) \in \mathcal{P}_{C_j}(\Delta)$ then the experimental result $fa = c$ is said to be an *anomalous* experimental result.

### 6.2.4. Falsifiability

For a database which gives competing inductive theories the practical question then arises as to which should be used. That is, which set of conjectures should we adopt as an extension to our belief set.

One well known approach to this question is to look at the *falsifiability* of each theory [**38**]. The theory is then stronger if it is more likely to be falsified. Einstein's theory of gravitation, the standard example of this, made many bold and observable predictions, opening it to being easily falsified were it incorrect.

We have a pragmatic use for this idea here. If our reasoner is faced with a number of competing theories then we would like to carry out experiments to decide between them, ultimately falsifying all but one. By taking the theory which is most falsifiable as a working theory we in a sense maximize the chance of this happening.

DEFINITION 6.9. Let $C$ be a set consistent with respect to a database $\Delta$. If $(fa = b) \in \mathcal{P}_C(\Delta)$ and the result of the experiment $fa$ is $fa = c$, $b \neq c$, then we say that $C$ is *falsified*.

Without any additional information about the domain, the chance of a theory being falsified according to this definition is thus proportional to the number of predictions it makes, and the quality of those predictions. A theory which makes no predictions can never be falsified and so is less useful than one which does make predictions and so can be tested. Furthermore a theory which makes, for example, a prediction about the result of a **sentence**-valued experiment is more likely to be right, and less likely to be falsified, than one making a prediction for a **person**-valued experiment. We capture these ideas in the following definition.

DEFINITION 6.10. Let $C_1$ and $C_2$ be two theories for a database $\Delta$. Then $C_1$ is *a stronger theory* then $C_2$ if

$$\left(1 - \prod_{e \in \mathcal{P}_{C_1}(\Delta)} \frac{1}{|\mathrm{cod}(e)|}\right) > \left(1 - \prod_{e \in \mathcal{P}_{C_2}(\Delta)} \frac{1}{|\mathrm{cod}(e)|}\right).$$

We frequently have conjectures only between predicates, in which case the the codomain consists of **True** and **False** and so has constant size 2. The above condition then becomes

$$\left(1 - \prod_{e \in \mathcal{P}_{C_1}(\Delta)} \frac{1}{2}\right) > \left(1 - \prod_{e \in \mathcal{P}_{C_2}(\Delta)} \frac{1}{2}\right),$$

$$\text{or} \quad (1/2)^{|\mathcal{P}_{C_1}(\Delta)|} < (1/2)^{|\mathcal{P}_{C_2}(\Delta)|},$$

giving the following simple result:

LEMMA 6.6. *Let $C_1$ and $C_2$ be two theories for a database $\Delta$ with every element of $C_1$ and $C_2$ a predicate. Then $C_1$ is a stronger theory than $C_2$ if*

$$|\mathcal{P}_{C_1}(\Delta)| > |\mathcal{P}_{C_2}(\Delta)|.$$

Our definition of theory strength is based on probabilities but there is one situation where we can be certain that a theory is stronger than another, captured in the following definition.

DEFINITION 6.11. A theory $C_1$ is *strictly stronger* than theory $C_2$ if any prediction of $C_2$ is also a prediction of $C_1$, and not conversely.

That is if $C_1$ is strictly stronger than $C_2$ then $\mathcal{P}_{C_2}(\Delta) \subset \mathcal{P}_{C_1}(\Delta)$ so that

$$\prod_{e \in \mathcal{P}_{C_2}(\Delta)} \frac{1}{|\mathrm{cod}(e)|} < \prod_{e \in \mathcal{P}_{C_1}(\Delta)} \frac{1}{|\mathrm{cod}(e)|},$$

giving the natural relationship between 'stronger' and 'strictly stronger':

LEMMA 6.7. *Let $C_1$ and $C_2$ be two theories for a database $\Delta$. If $C_1$ is strictly stronger than $C_2$ then $C_1$ is stronger than $C_2$.*

The following examples illustrate the calculation of falsifiability and show that the converse to this lemma does not hold.

EXAMPLE 6.7 (Predicate theory strength). Consider the three functions

$$\mathsf{male}, \mathsf{father}, \mathsf{tall} : \mathsf{person} \longrightarrow \mathsf{sentence}$$

for which we have a database about one generation of our family tree, given by

$$\Delta = \left\{ \begin{array}{l} \mathsf{father\ John} \longrightarrow \mathsf{True},\ \mathsf{father\ Peter} \longrightarrow \mathsf{True}, \\ \mathsf{male\ Peter} \longrightarrow \mathsf{True},\ \mathsf{male\ Mary} \longrightarrow \mathsf{False}, \\ \mathsf{tall\ John} \longrightarrow \mathsf{True},\ \mathsf{tall\ Mary} \longrightarrow \mathsf{True} \end{array} \right\}.$$

Induction results in the two maximally consistent sets of conjectures

$$C_1 = \{\mathsf{father} \simeq \mathsf{male},\ \mathsf{tall} \simeq \mathsf{True}\ !\},$$
$$C_2 = \{\mathsf{tall} \simeq \mathsf{father},\ \mathsf{father} \simeq \mathsf{True}\ !,\ \mathsf{tall} \simeq \mathsf{True}\ !\},$$

from which we obtain the predictions

$$\mathcal{P}_{C_1}(\Delta) = \left\{ \begin{array}{l} \mathsf{father\ Mary} \longrightarrow \mathsf{False},\ \mathsf{male\ John} \longrightarrow \mathsf{True}, \\ \mathsf{tall\ Peter} \longrightarrow \mathsf{True} \end{array} \right\},$$
$$\mathcal{P}_{C_2}(\Delta) = \left\{ \mathsf{father\ Mary} \longrightarrow \mathsf{True},\ \mathsf{tall\ Peter} \longrightarrow \mathsf{True} \right\}.$$

Hence we would say that $C_1$ is a stronger theory than $C_2$. Specifically, if Mary, John, and Peter are the only entities in **person** then for the three functions we have nine possible experiments, six of which we already have the results for in $\Delta$. If we supposed that $C_1$ was not a valid theory and that its predictions were simply random, then each of the remaining three experiments would support it with probability $\frac{1}{2}$. Thus the probability of it being falsified by the remaining experiments is $1 - (\frac{1}{2})^3 = 0.875$. On the other hand, the outcome of **male John** has no effect on $C_2$ and so its probability of being falsified is only $1 - (\frac{1}{2})^2 = 0.75$. We hence say that $C_1$ is stronger than $C_2$. $\diamond$

EXAMPLE 6.8 (Equal theory strength). The two maximally consistent $C_1$ and $C_2$ in Example 6.4 both give a single prediction and so we would say that they are *equally* strong. $\diamond$

EXAMPLE 6.9 (Multi-sorted theory strength). As an example which doesn't involve predicates, consider the following signature:

$$\Sigma = \left\{ \begin{array}{l} \mathsf{Alice, John, Jill, Jenny, Kelly,} \\ \mathsf{Paul, Peter, Mary} \in \mathsf{person,} \\ \mathsf{Blue, Brown, Green} \in \mathsf{colour,} \\ \mathsf{eyeColour : person} \rightarrow \mathsf{colour,} \\ \mathsf{father, mother, wife, sister : person} \rightarrow \mathsf{person} \end{array} \right\}.$$

We make the observations $\Delta$ about our families:

$$\Delta = \left\{ \begin{array}{l} \mathsf{father\ Alice} \rightarrow \mathsf{John, father\ Paul} \rightarrow \mathsf{Peter,} \\ \mathsf{wife\ John} \rightarrow \mathsf{Jill, wife\ Peter} \rightarrow \mathsf{Mary,} \\ \mathsf{mother\ Alice} \rightarrow \mathsf{Jill, sister\ John} \rightarrow \mathsf{Jenny,} \\ \mathsf{sister\ Jill} \rightarrow \mathsf{Mary, sister\ Mary} \rightarrow \mathsf{Kelly,} \\ \mathsf{eyeColour\ Alice} \rightarrow \mathsf{Blue, eyeColour\ John} \rightarrow \mathsf{Brown,} \\ \mathsf{eyeColour\ Jill} \rightarrow \mathsf{Blue, eyeColour\ Paul} \rightarrow \mathsf{Green,} \\ \mathsf{eyeColour\ Jenny} \rightarrow \mathsf{Brown} \end{array} \right\}.$$

We have three maximally consistent sets of conjectures

$$C_1 = \{\mathsf{eyeColour\ mother} \simeq \mathsf{eyeColour,\ wife\ father} \simeq \mathsf{mother}\},$$

$$C_2 = \{\mathsf{eyeColour\ sister} \simeq \mathsf{eyeColour,\ wife\ father} \simeq \mathsf{mother}\},$$

$$C_3 = \{\mathsf{eyeColour\ mother} \simeq \mathsf{eyeColour,\ eyeColour\ sister} \simeq \mathsf{eyeColour}\},$$

with corresponding predictions

$$\mathcal{P}_{C_1}(\Delta) = \{\mathsf{eyeColour\ Mary} \rightarrow \mathsf{Green,\ mother\ Paul} \rightarrow \mathsf{Mary}\},$$

$$\mathcal{P}_{C_2}(\Delta) = \left\{ \begin{array}{l} \mathsf{eyeColour\ Mary} \rightarrow \mathsf{Blue,\ eyeColour\ Kelly} \rightarrow \mathsf{Blue,} \\ \mathsf{mother\ Paul} \rightarrow \mathsf{Mary} \end{array} \right\},$$

$$\mathcal{P}_{C_3}(\Delta) = \{\mathsf{eyeColour\ Kelly} \rightarrow \mathsf{Blue,\ eyeColour\ Mary} \rightarrow \mathsf{Blue}\}.$$

With three elements in colour, the probability of $C_3$ being falsified is $1 - (\frac{1}{3})^2 = 0.889$. Eight elements in person gives the probability of $C_1$ being falsified as $1-(\frac{1}{3})(\frac{1}{8}) = 0.958$ and of $C_2$ being falsified as $1-(\frac{1}{3})^2(\frac{1}{8}) = 0.986$. We thus take $C_2$ as the strongest theory.

Note that $C_2$ is in fact strictly stronger than $C_3$, though only stronger than $C_1$. $\diamond$

Our considerations here, particularly in looking at the size of a conjecture's codomain, are similar to our numerical measures based on hypothesis testing and predictive strength (see Section 6.1). This kind of analysis is not available in standard logic programming where all statements are predicates and the notion of a codomain is not present. Treating the equations in the previous example in this way, we would not have been able to distinguish between the theories $C_1$ and $C_2$.

Finally, note that the problem of finding the theory of an arbitrary set of conjectures is likely to be computationally hard and a complete analysis of its complexity, in terms of the number of consistency checks required, is beyond the scope of this work. Our present interest is in the reasoning itself, rather than in the practicalities of efficient implementation. However, obtaining a single maximally consistent set is linear in the number of consistency checks required since any conjecture can be extended to a maximal set by successively adding conjectures which retain consistency until all conjectures have been tried. That is, with $N$ conjectures we can find a maximally consistent set with $N$ tests for consistency. Thus for large systems of conjectures we can also make use of our falsifiability criterion by taking a small sample of maximally consistent sets, generated by extending each of a collection of conjectures, and selecting the strongest of those as the working theory. Choosing the conjectures to extend from those not already in a generated maximal set can result in a good approximation to the full theory of the conjectures.

## 6.3. Blocks World

The well-known blocks world ([43], [21]) provides us with a graphical illustration of our reasoner's inductive beliefs. Rather than questioning the reasoner to find out what it believes, as in previous dialogues, at each stage now the reasoner draws a picture of its beliefs to show what it thinks.

We represent the blocks world by the signature

$$\Sigma = \left\{ \begin{array}{l} \mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}, \mathsf{E} \in \mathsf{block}, \\ \mathsf{below}, \mathsf{above} : \mathsf{block} \rightarrow \mathsf{block}, \\ \mathsf{table} : \mathsf{block} \rightarrow \mathsf{sentence} \end{array} \right\}.$$

For example, we use $\mathsf{below}\ \mathsf{B} = \mathsf{A}$ to say that A is below B and $\mathsf{table}\ \mathsf{A} = \mathsf{True}$ to say that A is on the table.

The reasoner uses values of **table** and **below** to draw a picture of its beliefs. (We have deliberately not used **above** since it leaves the reasoner to figure out **below** values from given information about **above**. Note also that using **above** for table-up building requires the solution of equational systems, as described in Section 3.7.) For each block A, B, C, D, E, the reasoner first consults its declared knowledge to determine if the block is on the table or on a block which it knows the location of. If successful it draws a block in its place in the world. If the declared knowledge is not sufficient then it tries to determine the same information using its inductive conjectures. If this is successful the block is also drawn, but is shaded to indicate it is based on conjectured information. If no information is available then the block is not shown.

EXAMPLE 6.10 (Simple learning). Figure 6.8 illustrates this process with a simple sequence of increasing information.

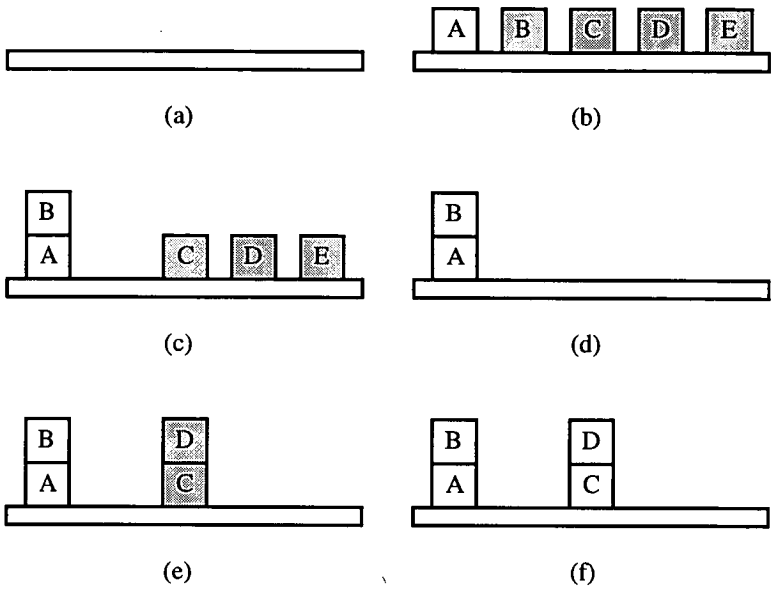



FIGURE 6.8. Growing knowledge in a blocks world

The reasoner begins with no knowledge about the position of the blocks, giving the empty table in Figure 6.8(a). The remaining arrangements are the result of the following statements.

(b) 'A is on the table.' The reasoner makes the conjecture **table** $\simeq$ **True** !, guessing that this property holds for all blocks. A is drawn with certainty on the table, with the remaining blocks also positioned on the table but shaded to indicate that their position is only a conjecture.
(c) 'A is below B.' The reasoner makes the additional conjecture **table below** $\simeq$ **True** !, that everything below something is on the table. However, this conjecture doesn't provide any new information.
(d) 'B is not on the table.' The initial conjecture that everything is on the table is falsified and blocks C, D, and E are removed.
(e) 'C is below D.' The reasoner now knows where D is in relation to C but doesn't yet know the location of C. However from earlier information it is able to conjecture that C is on the table, so placing both C and D.
(f) 'C is on the table.' Complete information is now available about the position of C and hence of D.

This was the first example tried after implementing this blocks world reasoner. We were quite surprised when C and D appeared in Figure 6.8(e), an encouraging sign that the reasoner can out-reason its creator. $\diamond$

EXAMPLE 6.11 (Working theories). The world drawn in Figure 6.9 introduces the **above** function and in doing so makes necessary consistency analysis to find a strongest working theory.

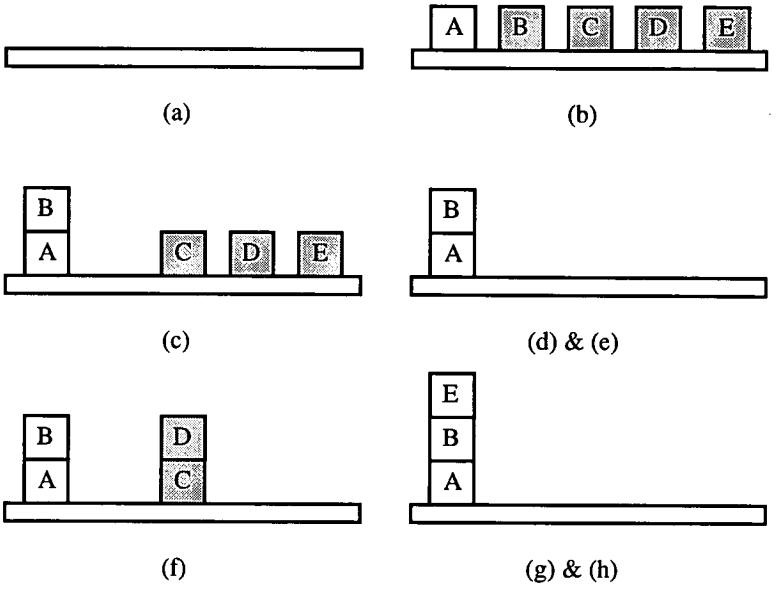



FIGURE 6.9. A blocks world with multiple theories

The first three statements are the same as in the previous example: 'A is on the table,' 'A is below B,' and 'B is not on the table.' Figure 6.9(e) is the result of stating 'B is above A.' Physically, this tells us nothing new since we already had **below B = A**. However, the four conjectures generated are inconsistent, resulting in the maximally consistent subsets given in Table 6.1. None give any predictions and so each is unfalsifiable, leaving us unable to choose a strongest theory. We arbitrarily choose the first as a working theory. In our blocks world this is in fact redundant: with no predictions we will not see any results of the reasoner's conjectures.

| Conjectures | Predictions | Falsifiability |
|---|---|---|
| **below above ≃ i**<br>**above below ≃ i** | $\phi$ | 0.0 |
| **below above ≃ i**<br>**table above ≃ False !** | $\phi$ | 0.0 |
| **above below ≃ True !**<br>**table below ≃ True !** | $\phi$ | 0.0 |
| **table below ≃ True !**<br>**table above ≃ False !** | $\phi$ | 0.0 |

TABLE 6.1. Maximally consistent sets for Figure 6.9(e)

Figure 6.9(f) incorporates 'C is below D,' resulting in the same maximally consistent sets but this time with associated predictions, as shown in Table 6.2. The third theory is most falsifiable and so we choose it as a working theory. The prediction table C = True allows the reasoner to conjecture the positions of C and D on the table. Even though it knows D is to be drawn on top of C, D is shaded since it is uncertain where C is.

| Conjectures | Predictions | Falsifiability |
|---|---|---|
| below above $\simeq$ i<br>above below $\simeq$ i | above C = D | 0.8 |
| below above $\simeq$ i<br>table above $\simeq$ False ! | $\phi$ | 0.0 |
| above below $\simeq$ True !<br>table below $\simeq$ True ! | table C = True<br>above C = D | 0.9 |
| table below $\simeq$ True !<br>table above $\simeq$ False ! | table C = True | 0.5 |

TABLE 6.2. Maximally consistent sets for Figure 6.9(f)

The approximate equivalence between falsifiability and the number of predictions has an obvious practical meaning here. If more predictions are made then the reasoner will be able to describe the location of more of the blocks so that the strongest theory will be the best at completing the reasoner's world.

In Figure 6.9(g) we have added that 'B is below E', falsifying two of the previous competing theories. The remaining two and their predictions are given in Table 6.3. Note that we lose the prediction table C = True and hence can no longer show C and D, even though we know C is below D.

| Conjectures | Predictions | Falsifiability |
|---|---|---|
| below above $\simeq$ i<br>above below $\simeq$ i | above B = E<br>above C = D | 0.96 |
| below above $\simeq$ i<br>table above $\simeq$ False ! | $\phi$ | 0.0 |

TABLE 6.3. Maximally consistent sets for Figure 6.9(g)

The remaining dilemma here is quite subtle. The three conjectures generated all seem natural and so it is strange that there is an inconsistency. The problem lies in the equation above below $\simeq$ i which in fact is *not* valid in the physical world we are modelling. Applying below to A can never give a block and so we can never obtain the right-hand identity. To capture this we add

that 'There is nothing below A,' encoded in the equation **below** $\mathsf{A} = \phi$. We then have the unique maximal theory given in Table 6.4. $\diamond$

| Conjectures | Predictions | Falsifiability |
|---|---|---|
| **below above** $\simeq$ **i** | $\phi$ | 0.0 |
| **table above** $\simeq$ **False !** | | |

TABLE 6.4. Unique maximally consistent set for Figure 6.9(h)

## 6.4. Theoretical Systems

### 6.4.1. Data Compression and Axioms

One important application of summative knowledge, and a very important motivation of inductive learning, is to reduce the amount of data that needs to be known. For example, in finding a law that governs the motion of a planet around the sun we can then discard most of our observations of it's motion since they can be generated from the law and some initial conditions.

We might call this process 'inverse deduction'. In contrast to Knuth-Bendix completion which deduces all possible data from a given equational system, here we want to find a minimal equational system from which we could deduce all known data. The obvious inference rule for this process then is simply the inverse of the *Deduce* rule used by KB:

$$\textit{Compress} \quad R \cup \{s \to t\} \quad \Rightarrow \quad R \quad \text{if } s \leftarrow_{R\setminus(s\to t)} u \rightarrow_{R\setminus(s\to t)} t$$

This single inference rule applied successively will take an initial rewrite system $R_0$ and remove all rules that are deductive consequences of remaining rules. If $R_0$ is finite then this procedure must always terminate with some $R_n$, since a rule is removed at each step of the algorithm. The initial $R_0$ will be the known database $\Delta$ together with the canonical form of the accepted facts and conjectures from IND($\Delta$).

This process is identical to the task of finding a minimal system for inequations, examined in Section 3.4.4. As we saw there, the result of this process can be effected by the ordering of the rules in $R$ and so we again refine the process. Specifically, $(s_1 \to t_1) > (s_2 \to t_2)$ if and only if $s_1 > s_2$, or $s_1 \equiv s_2$ and $t_1 > t_2$. We then have

$$\begin{aligned}\textit{Compress} \quad R \cup \{s \to t\} \quad \Rightarrow \quad R \quad &\text{if } s \leftarrow_{R\setminus(s\to t)} u \rightarrow_{R\setminus(s\to t)} t, \\ &s \to t \text{ examined from highest to lowest.}\end{aligned}$$

The result for a given system is then uniquely determined. We could follow the development of minimal systems and provide a procedure whereby two

equivalent systems would have the same compressed form. This is straightforward but unnecessary here since we are only looking at isolated systems.

This compression algorithm can also be approached from the viewpoint of theoretical systems. Popper [**39**] defines a set of *axioms* for a system to be a set satisfying the following properties:

1. The axioms must be consistent;
2. No axiom should be the consequence of other axioms;
3. Every statement in the system should be a consequence of the axioms;
4. Every axiom is necessary for describing the system (equivalently, there are no consequences of the axioms that are not in the original system).

It is clear that if the conjectures are consistent with respect to $\Delta$ then the result of the compression procedure will satisfy all of these conditions with respect to the closure of $\Delta$. Thus we call the final set of rules $R_n$ with $\rightarrow$ replaced by $=$ a set of *axioms* for the closure of $\Delta$. If $\mathrm{IND}(\Delta)$ contains any conjectures, rather than just summative facts, then we more specifically say it is a set of *conjectured axioms*

EXAMPLE 6.12 (Conjectured axioms). Consider the database $\Delta_2$ in Example 5.1 from which we conjectured that male father $\simeq$ True !. To find the axioms for $\Delta$ we apply the above compression procedure to

$$R_0 = \left\{ \begin{array}{l} \text{father Paul} \rightarrow \text{Peter, father Alice} \rightarrow \text{John,} \\ \text{male Peter} \rightarrow \text{True, male John} \rightarrow \text{True,} \\ \text{male Alice} \rightarrow \text{False, male father} \rightarrow \text{True !} \end{array} \right\}.$$

The two data involving male are each the consequence of the corresponding father information together with the conjecture relating male to father. This gives the conjectured axioms

$$R = \left\{ \begin{array}{l} \text{father Paul} = \text{Peter, father Alice} = \text{John,} \\ \text{male Alice} = \text{False, male father} = \text{True !} \end{array} \right\}.$$

◇

Note that we also have a relationship here with the theory of prime implicants [**40**]. Viewing the $R_0$ and $R$ above as conjunctions of equations, $R$ is then a prime implicant of $R_0$ since it is equivalent to $R_0$ but would cease to be so if any further equations were removed from it.

Finally, if the conjectures $E$ are not consistent with respect to the data $\Delta$ then we cannot axiomatize the closure of $E$ with respect to $\Delta$. However we can still attempt to find a compressed system. Suppose in the above example that our conjecture was that male father $\simeq$ False !. Then during compression we would have the critical pair

$$\text{male Peter} \longleftarrow \text{male father Paul} \longrightarrow \text{False}$$

which does not match the known datum male Peter $\rightarrow$ True and so *Compress* cannot be applied. Thus an interesting empirical measure of the consistency

of a set of conjectures with a database is how well its canonical form compresses that database.

### 6.4.2. Toy Physics

Having described the induction procedure in detail, we now give an illustration of it in terms of scientific discovery. Consider a toy world consisting of a square with vertices A, B, C, D. We are interested in the behaviour of two physical processes, h and r, which can be applied to the square, moving each vertex into a new position. The only experiment we can perform is to focus on a particular vertex, apply one of the processes, and record which vertex ours replaces.

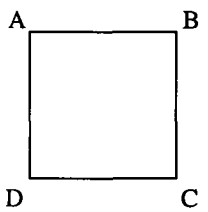



FIGURE 6.10. A simple physical system

To model this world we use the signature

$$\Sigma = \left\{ \begin{array}{l} \mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D} \in \mathsf{point}, \\ \mathsf{h}, \mathsf{r} : \mathsf{point} \rightarrow \mathsf{point} \end{array} \right\}$$

with entities $\{\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}\}$.

### 6.4.3. Theories Concerning Points

We begin with two preliminary experiments to try and determine the behaviour of h, giving an initial database

$$D = \{\mathsf{h}\ \mathsf{A} \rightarrow \mathsf{D},\ \mathsf{h}\ \mathsf{B} \rightarrow \mathsf{C}\}.$$

$\mathrm{IND}(D)$ is empty for all function lengths and so we cannot make any conjectures. We turn our attention to r, making observations for the same entities to give the extended

$$D_1 = D \cup \{\mathsf{r}\ \mathsf{A} \rightarrow \mathsf{B},\ \mathsf{r}\ \mathsf{B} \rightarrow \mathsf{C}\}.$$

Now we are able to make our first conjecture,

$$\mathrm{IND}(D_1) = \{\mathsf{r}\ \mathsf{r} \simeq \mathsf{h}\ \mathsf{r}\},$$

since $\mathsf{r}\ \mathsf{r}\ \mathsf{A} \rightarrow \mathsf{C} \leftarrow \mathsf{h}\ \mathsf{r}\ \mathsf{A}$. The term ordering chosen by the reasoner has $\mathsf{h} < \mathsf{r}$, trying to reduce new terms encountered, such as r, to known terms, in this case h. $\mathrm{KBI}(D_1 \cup (\mathsf{r}\ \mathsf{r} = \mathsf{h}\ \mathsf{r}))$ is then unbounded, containing $\mathsf{r}\ \mathsf{h}^k\mathsf{C} \rightarrow \mathsf{h}^{k+1}\mathsf{C}$

for all $k \geq 1$, but we are still able to use proof by contradiction to look for predictions, in this case finding none.

A further observation gives $D_2 = D_1 \cup \{\mathsf{r\ C} \rightarrow \mathsf{D}\}$, resulting in the larger

$$\mathrm{IND}(D_2) = \left\{ \begin{array}{l} \mathsf{r\ r} \simeq \mathsf{h\ r},\ \mathsf{r\ r} \simeq \mathsf{r\ h},\ \mathsf{r\ r\ r} \simeq \mathsf{h}, \\ \mathsf{r\ h\ r} \simeq \mathsf{h},\ \mathsf{r\ r\ r} \simeq \mathsf{r\ h\ r} \end{array} \right\},$$

and giving our first predictions

$$\mathcal{P}(D_2) = \{\mathsf{h\ C} = \mathsf{D},\ \mathsf{h\ D} = \mathsf{C},\ \mathsf{r\ D} = \mathsf{C}\}.$$

It is interesting to note, from a scientific viewpoint, how additional observations about one physical process can allow us to make predictions about the behaviour of another process.

Continuing this path we make the final possible observation for r, with $D_3 = D_2 \cup \{\mathsf{r\ D} \rightarrow \mathsf{A}\}$. This falsifies the previous theory and gives

$$\Delta_3 = \mathrm{IND}(D_3, 3) = \left\{ \begin{array}{l} \mathsf{r\ r\ r} \simeq \mathsf{h\ r\ h},\ \mathsf{r\ h\ r} \simeq \mathsf{h}, \\ \mathsf{h\ r\ h} \simeq \mathsf{h},\ \mathsf{r\ h\ r} \simeq \mathsf{h\ r\ h} \end{array} \right\}.$$

There are now also conjectures of term length 4 to consider, including the fact $\mathsf{r\ r\ r\ r} = \mathsf{i}$, but we firstly see what we can say from the restriction to length 3. $\Delta_3$ is inconsistent with the data; Table 6.5 shows the analysis of $\Delta_3$ in terms of maximally consistent subsets.

| Conjectures | Predictions | Falsifiability |
|---|---|---|
| $\mathsf{r\ r\ r} \simeq \mathsf{h\ r\ h}$ | $\mathsf{h\ C} = \mathsf{B}$ | 0.9375 |
| $\mathsf{r\ h\ r} \simeq \mathsf{h}$ | $\mathsf{h\ D} = \mathsf{A}$ | |
| $\mathsf{h\ r\ h} \simeq \mathsf{h}$ | $\mathsf{h\ D} = \mathsf{C}$ | 0.75 |
| $\mathsf{r\ h\ r} \simeq \mathsf{h\ r\ h}$ | $\phi$ | 0.0 |

TABLE 6.5. Maximally consistent sets from $D_3$

Two of the competing theories again give predictions about process h from observations of process r, the strongest theory giving predictions for all applications of $h$. Taking this $C = \{\mathsf{r\ r\ r} \simeq \mathsf{h\ r\ h},\ \mathsf{r\ h\ r} \simeq \mathsf{h}\}$ as a working theory we have the axiomatization

$$\left\{ \begin{array}{l} \mathsf{h\ B} = \mathsf{C},\ \mathsf{r\ A} = \mathsf{B},\ \mathsf{r\ B} = \mathsf{C},\ \mathsf{r\ C} = \mathsf{D},\ \mathsf{r\ D} = \mathsf{A}, \\ \mathsf{r\ r\ r} = \mathsf{h\ r\ h},\ \mathsf{r\ h\ r} = \mathsf{h} \end{array} \right\},$$

dropping the observation $\mathsf{h\ A} \rightarrow \mathsf{D}$. This introduces an alternative notion of theory strength, since adopting $\{\mathsf{h\ r\ h} \simeq \mathsf{h}\}$ instead of $C$ results in no compression of the data. Rather than looking at the predictive power of a theory, we could say a theory is stronger if it better explains the existing data, measured here by the degree of compression it achieves. These definitions are distinct; for example, the theories $\{\mathsf{h\ r\ h} \simeq \mathsf{h}\}$ and $\{\mathsf{r\ h\ r} \simeq \mathsf{h\ r\ h}\}$ both give zero compression, but based on predictions we would say that

$\{\mathsf{h\ r\ h} \simeq \mathsf{h}\}$ is stronger than $\{\mathsf{r\ h\ r} \simeq \mathsf{h\ r\ h}\}$. Since our main application of induction lies in prediction, it makes sense for us to use this latter measure.

To avoid considering all 16 conjectures of length 4 it would be useful to know if the fact $\mathsf{r\ r\ r\ r} = \mathsf{i}$ is a consequence of our conjectured theory $C$. Applying proof by contradiction to the possible theorem

$$(\{\mathsf{r\ r\ r} \simeq \mathsf{h\ r\ h},\ \mathsf{r\ h\ r} \simeq \mathsf{h}\}, \mathsf{r\ r\ r\ r} = \mathsf{i})$$

shows it is not a consequence, but has the single side condition $\mathsf{h\ h} = \mathsf{i}$. The predictions of this condition, $\{\mathsf{h\ C} = \mathsf{B},\ \mathsf{h\ D} = \mathsf{A}\}$, are the same as those for $C$ and so if we verify these predictions for $C$ we are supporting $\mathsf{r\ r\ r\ r} = \mathsf{i}$ as a consequence of our working theory.

Returning then to $\mathsf{h}$, the last two observations indeed match these predictions, giving $D_4 = D_3 \cup \{\mathsf{h\ C} = \mathsf{B},\ \mathsf{h\ D} = \mathsf{A}\}$. With all possible applications observed, any induced equation will now be a fact. For term lengths of at most 3 we have $C' = \mathrm{IND}(D_4, 3)$ consisting of 9 facts. These are consistent with $D_4$ and include $\mathsf{h\ h} = \mathsf{i}$ so that $\mathsf{r\ r\ r\ r} = \mathsf{i}$ is a consequence of $C'$. In fact all 44 facts of length 4 are consequences of $C'$ by invariance, since

$$\mathrm{KB}(C') = \mathrm{KB}(\mathrm{IND}(D_4, 4)) = \left\{ \begin{array}{l} \mathsf{h\ h} \to \mathsf{i},\ \mathsf{r\ r\ h} \to \mathsf{h\ r\ r}, \\ \mathsf{r\ r\ r} \to \mathsf{h\ r\ h},\ \mathsf{r\ h\ r} \to \mathsf{h} \end{array} \right\}.$$

Our final axiomatization is

$$\left\{ \begin{array}{l} \mathsf{h\ D} = \mathsf{A},\ \mathsf{r\ B} = \mathsf{C},\ \mathsf{r\ C} = \mathsf{D},\ \mathsf{r\ D} = \mathsf{A}, \\ \mathsf{h\ h} = \mathsf{i},\ \mathsf{r\ r\ r} = \mathsf{h\ r\ h},\ \mathsf{r\ h\ r} = \mathsf{h} \end{array} \right\},$$

eliminating half of the eight observations.

### 6.4.4. Theories Concerning Numbers

Having completed our study of the processes $\mathsf{h}$ and $\mathsf{r}$ on points, we now turn our attention to a new domain. We extend our language to a signature

$$\Sigma' = \Sigma \cup \left\{ \begin{array}{l} 1, 2, 3, \ldots \in \mathsf{number}, \\ \mathsf{succ}, \mathsf{double} : \mathsf{number} \to \mathsf{number}, \\ \mathsf{odd}, \mathsf{even} : \mathsf{number} \to \mathsf{sentence} \end{array} \right\}.$$

Whereas **point** contained only a finite number of entities, here we have an infinite collection from which to choose the objects of our experiments. This means from the start that we will be unable to obtain any facts from a finite database of observations. (Compare this to the discussion following Example 3.13 where a Peano scheme of constructors allowed the finite description of infinite databases and hence the establishment of facts).

We begin by making a complete series of observations for the processes **succ**, **odd**, and **even** when applied to the sample **1**, **2**, **3**, and **4** from **number**. This

gives the database

$$N = D_4 \cup \left\{ \begin{array}{l} \mathsf{succ}\ 1 = 2,\ \mathsf{odd}\ 1 = \mathsf{True},\ \mathsf{even}\ 1 = \mathsf{False}, \\ \mathsf{succ}\ 2 = 3,\ \mathsf{odd}\ 2 = \mathsf{False},\ \mathsf{even}\ 2 = \mathsf{True}, \\ \mathsf{succ}\ 3 = 4,\ \mathsf{odd}\ 3 = \mathsf{True},\ \mathsf{even}\ 3 = \mathsf{False}, \\ \mathsf{succ}\ 4 = 5,\ \mathsf{odd}\ 4 = \mathsf{False},\ \mathsf{even}\ 4 = \mathsf{True} \end{array} \right\},$$

augmenting the earlier observations in $D_4$. The extension of $\Sigma$ to $\Sigma'$ satisfies the conditions of Lemma 5.3 and so we are guaranteed that the facts obtained for $D_4$ will remain facts for the extended $N$. Furthermore $N \subseteq \mathcal{F}_{\Sigma''}$, where

$$\Sigma'' = \Sigma \cup \left\{ \begin{array}{l} 1, 2, \ldots, 8 \in \mathsf{number}, \\ \mathsf{succ}, \mathsf{double} : \mathsf{number} \rightarrow \mathsf{number}, \\ \mathsf{odd}, \mathsf{even} : \mathsf{number} \rightarrow \mathsf{sentence} \end{array} \right\}$$

is a finite restriction of $\Sigma'$. By Lemma 5.4 we can thus treat $N$ as having signature $\Sigma''$ for the purposes of induction, guaranteed that any conjectures we obtain will be conjectures for $N$ with the original $\Sigma'$. Here we find

$$\Delta_2 = \mathrm{IND}(N, 2) = \{\mathsf{even}\ \mathsf{succ} \simeq \mathsf{odd},\ \mathsf{odd}\ \mathsf{succ} \simeq \mathsf{even}\}.$$

With these we obtain the conjectured axiomatization

$$\left\{ \begin{array}{l} \mathsf{even}\ \mathsf{succ} = \mathsf{odd},\ \mathsf{odd}\ \mathsf{succ} = \mathsf{even}, \\ \mathsf{succ}\ 1 = 2,\ \mathsf{succ}\ 2 = 3, \\ \mathsf{succ}\ 3 = 4,\ \mathsf{succ}\ 4 = 5, \\ \mathsf{odd}\ 1 = \mathsf{True},\ \mathsf{odd}\ 2 = \mathsf{False} \end{array} \right\}.$$

Turning to the process **double**, a single observation gives

$$N_1 = N \cup \{\mathsf{double}\ 1 = 2\}.$$

There are many conjectures of term length 2 over $\Sigma''$, summarized in the consistency analysis of Table 6.6. Only one theory makes predictions about the behaviour of **double**, in addition to predictions for **odd** and **even**, and is thus the natural choice for the working theory in terms of falsifiability.

There are no crucial experiments and so we take for our next observation an experiment corresponding to one of the predictions of the working theory. This gives

$$N_2 = N_1 \cup \{\mathsf{double}\ 2 = 4\},$$

with

$$\Delta_2 = \mathrm{IND}(N_2, 2) = \left\{ \begin{array}{l} \mathsf{even}\ \mathsf{succ} \simeq \mathsf{odd},\ \mathsf{odd}\ \mathsf{succ} \simeq \mathsf{even}, \\ \mathsf{even}\ \mathsf{double} \simeq \mathsf{True}\ !,\ \mathsf{odd}\ \mathsf{double} \simeq \mathsf{False}\ !, \\ \mathsf{double}\ \mathsf{double} \simeq \mathsf{double}\ \mathsf{succ} \end{array} \right\}.$$

With **double** $>$ **succ**, $\mathrm{KBI}(N_2 \cup \Delta_2)$ is unbounded, containing a rewrite rule $\mathsf{double}\ \mathsf{succ}^k 5 \rightarrow \mathsf{double}\ 3$ for each $k \geq 1$. As with proof by contradiction we halted KBI before obtaining $\perp$, suspecting the result was unbounded. We

thus assume that $\Delta_2$ is consistent with $N_2$. We no longer get predictions about **double**, leaving

$$\mathcal{P}(N_2) = \{\textsf{even 5 = False, odd 5 = True}\}.$$

Thus we have no guidance in choosing the next experiment for **double**, so we apply it to an arbitrary entity to give

$$N_3 = N_2 \cup \{\textsf{double 3 = 6}\}.$$

This gives the same conjectures of length 2 as for $N_2$, but now there are many predictions:

$$\mathcal{P}(N_3) = \left\{ \begin{array}{l} \textsf{double 4 = 6, double 5 = 6, double 6 = 6,} \\ \textsf{even 5 = False, odd 5 = True,} \\ \textsf{even 6 = True, odd 6 = False} \end{array} \right\}.$$

Here we begin to see the scientific rewards of studying the process **double**. Our observations of it now provide new information about the original processes **odd** and **even**. For this particular example it is interesting to note that the new information is correct even though the predictions make for **double** itself are not.

Emboldened by this discovery, we make the final in our series of observations of **double** (this time suggested by the predictions from $N_3$), giving

$$N_4 = N_3 \cup \{\textsf{double 4 = 8}\}.$$

The conjectures of length 2 are now

$$\Delta_2 = \mathrm{IND}(N_4, 2) = \left\{ \begin{array}{l} \textsf{even succ} \simeq \textsf{odd, odd succ} \simeq \textsf{even,} \\ \textsf{even double} \simeq \textsf{True !, odd double} \simeq \textsf{False !} \end{array} \right\},$$

with final predictions

$$\mathcal{P}(N_4) = \left\{ \begin{array}{l} \textsf{even 5 = False, odd 5 = True,} \\ \textsf{even 6 = True, odd 6 = False,} \\ \textsf{even 8 = True, odd 8 = False} \end{array} \right\}.$$

Again, observations of **double** have given new information about the earlier processes. It is not surprising then that we obtain an axiomatization of the closure which is free of the original data for **odd** and **even**:

$$\left\{ \begin{array}{l} \textsf{even succ = odd, odd succ = even,} \\ \textsf{even double = True !, odd double = False !,} \\ \textsf{succ 1 = 2, succ 2 = 3,} \\ \textsf{succ 3 = 4, succ 4 = 5,} \\ \textsf{double 1 = 2, double 2 = 4,} \\ \textsf{double 3 = 6, double 4 = 8} \end{array} \right\}.$$

Here we have only looked at conjectures of maximum length 2. For $N_4$ there are 50 conjectures of length 3, inconsistent as a whole with the data. As in science, when looking at the more complicated conjectures it is generally necessary to make more observations to obtain a single consistent theory.

| Conjectures | Predictions | Falsifiability |
|---|---|---|
| double $\simeq$ succ | double 2 = 3 | 0.998 |
| even succ $\simeq$ odd | double 3 = 4 | |
| odd succ $\simeq$ even | double 4 = 5 | |
| even double $\simeq$ odd | even 5 = False | |
| even double $\simeq$ even succ | odd 5 = True | |
| odd double $\simeq$ even | | |
| odd double $\simeq$ odd succ | | |
| succ double $\simeq$ succ succ | | |
| even succ $\simeq$ odd | even 5 = False | 0.75 |
| odd succ $\simeq$ even | odd 5 = True | |
| even double $\simeq$ True ! | | |
| odd double $\simeq$ False ! | | |
| even succ $\simeq$ odd | even 5 = False | 0.75 |
| odd succ $\simeq$ even | odd 5 = True | |
| even double $\simeq$ True ! | | |
| odd double $\simeq$ even | | |
| odd double $\simeq$ odd succ | | |
| even succ $\simeq$ odd | even 5 = False | 0.75 |
| odd succ $\simeq$ even | odd 5 = True | |
| even double $\simeq$ odd | | |
| even double $\simeq$ even succ | | |
| odd double $\simeq$ False ! | | |
| even succ $\simeq$ odd | even 5 = False | 0.5 |
| even double $\simeq$ True ! | | |
| odd double $\simeq$ odd succ | | |
| succ double $\simeq$ succ succ | | |
| odd succ $\simeq$ even | odd 5 = True | 0.5 |
| even double $\simeq$ even succ | | |
| odd double $\simeq$ False ! | | |
| succ double $\simeq$ succ succ | | |
| even double $\simeq$ True ! | $\phi$ | 0.0 |
| odd double $\simeq$ False ! | | |
| succ double $\simeq$ succ succ | | |
| even double $\simeq$ True ! | $\phi$ | 0.0 |
| odd double $\simeq$ even | | |
| succ double $\simeq$ succ succ | | |
| even double $\simeq$ odd | $\phi$ | 0.0 |
| odd double $\simeq$ False ! | | |
| succ double $\simeq$ succ succ | | |

TABLE 6.6. Maximally consistent sets from $N_1$

## 6.5. Reasoning with Conjectures

So far we have looked at the process of making conjectures from a database of information, and at how we might choose from these a conjectured belief set, a working theory that is consistent with the database.

The next issue then is how we should reason with these extended beliefs. The simplest approach is to accept all conjectures $f \simeq g$ as equations $f = g$, giving a standard system for which we can use the deductive reasoning methods of Chapter 3. Of course this is a very large leap of faith, assigning the tentative knowledge the same strength of belief as our ground observations.

### 6.5.1. Modal Operators

Ideally when reasoning with both declared and conjectured knowledge we should be aware of the role the tentative information has in our deductions. Specifically, if we try proving a possible theorem with hypotheses containing both equations and conjectures then we want to know if any of the conjectures are involved in the proof (or in generating side conditions). If we have a theorem $F \Rightarrow f$ involving conjectures in its proof we might then say that $(F, f)$ is *probably* a theorem, or that $F \Rightarrow f$ is probably true.

Here we have an obvious connection with modal logic. The standard *possibility* operator $\Diamond$ can be applied to an equation $f = g$ to indicate the equation is possible, $\Diamond(f = g)$. By possible we mean that $f = g$ is not *impossible*. For our equational reasoning we would say an equation $f = g$ is impossible for some system $F$ if $F \Rightarrow (f \neq g)$. Thus

$$F \Rightarrow \Diamond(f = g) \text{ if and only if } F \not\Rightarrow (f \neq g).$$

However the notion of possibility is weaker than our notion of conjecture. A conjecture is certainly possible, by definition, since it can admit no falsifying instances. Yet a possible equation need not be a conjecture since we additionally require a conjecture to have a least one validating instance.

This has obvious parallels in science. For any situation a scientist is faced by an infinite number of *possible* theories. The natural first step in finding the 'best' theory is to look only at those theories which are suggested by observations. As shown in Theorem 5.3, for our world of term equations this then reduces the task to examining a finite collection of conjectures.

We have also seen, in Section 6.1.3, that a possible equation which is not a conjecture essentially contains no information. The probability $P_r$ of a conjecture based on $r$ supporting observations holding when applied to another entity was

$$P_r = \frac{r+1}{r+2}.$$

Without any confirming observations we have $P_0 = \frac{1}{2}$, so that the possible equation is equally likely to be right or wrong when applied to any new entity in its domain. Because of this we can have no belief in its predictions, making it worthless as a conjecture.

To discuss our conjectures we thus need a new modal operator. Define $\triangle$ to be the *probability* operator (not to be confused with the database notation $\Delta$), and read $F \Rightarrow \triangle(f = g)$ as it is *probable* that $f = g$ under $F$, or that $F \Rightarrow (f = g)$ is probably true. Our collection of equations that are probably true will contain the conjectures from $F$. That is,

$$\text{if } (f \simeq g) \in \text{IND}(F) \text{ then } F \Rightarrow \triangle(f = g).$$

We can use the consequence relation $\Rightarrow$ to deduce additional probable theorems from this initial set, as we will see in the next section.

Facts give an even stronger sense of probability, and we can similarly define

$$F \Rightarrow \triangle^*(f = g) \text{ if } (f = g) \in \text{IND}(F)$$

and read $F \Rightarrow \triangle^*(f = g)$ as that $F \Rightarrow (f = g)$ is *certainly* true.

The final operator worth defining is the standard *necessity* operator $\Box$, which we will view by reading $F \Rightarrow \Box(f = g)$ as $F \Rightarrow (f = g)$ is *definitely* true. We use 'definitely' to reflect that $(f = g)$ is a deductive consequence of the *definitions* captured in $F$. That is,

$$F \Rightarrow \Box(f = g) \text{ if and only if } F \Rightarrow (f = g).$$

Note that we have the standard equivalences

$$\Box(f = g) \equiv \neg\Diamond(f \neq g),$$
$$\Diamond(f = g) \equiv \neg\Box(f \neq g).$$

If we write $F_S = \{f = g \mid F \Rightarrow S(f = g)\}$ for any modal operator $S$ then we have the sequence

$$F_\Box \subseteq F_{\triangle^*} \subseteq F_\triangle \subseteq F_\Diamond.$$

### 6.5.2. Reasoning with Probability

DEFINITION 6.12. Let $\triangle F = \{f = g \mid f \simeq g \in \text{IND}(F)\}$. We say that $f = g$ is a *probable theorem* of $F$ if $F \cup \triangle F \Rightarrow (f = g)$ and $F \not\Rightarrow (f = g)$. If $f = g$ is a probable theorem of $F$ we write $F \Rightarrow \triangle(f = g)$.

The condition that $f = g$ is not already a deductive consequence of $F$ is reminiscent of our definition of a side condition being non-extraneous. Here we might similarly say that $F \Rightarrow \triangle(f = g)$ is an *extraneous* probable theorem if in fact $F \Rightarrow (f = g)$.

EXAMPLE 6.13 (Probable truth). Consider the database $D_1$ in Section 6.4.2, given by

$$D_1 = \{\mathsf{h\ A} \rightarrow \mathsf{D},\ \mathsf{h\ B} \rightarrow \mathsf{C},\ \mathsf{r\ A} \rightarrow \mathsf{B},\ \mathsf{r\ B} \rightarrow \mathsf{C}\}.$$

We found that $\triangle D_1 = \{\mathsf{r\ r} = \mathsf{h\ r}\}$, so we have, for example,

$$D_1 \Rightarrow \triangle(\mathsf{r\ r\ r} = \mathsf{r\ h\ r}),$$
$$D_1 \Rightarrow \triangle(\mathsf{r\ r\ C} = \mathsf{h\ r\ C}), \text{ and}$$
$$D_1 \not\Rightarrow \triangle(\mathsf{r\ r\ A} = \mathsf{h\ r\ A}),$$

since we already have $D_1 \Rightarrow (\mathsf{r\ r\ A} = \mathsf{h\ r\ A})$. $\diamond$

EXAMPLE 6.14 (Probable answers). In this example we give a better indication of how we implement the probable reasoning. Suppose we have beliefs

$$F = \left\{ \begin{array}{l} \mathsf{father\ Alice} = \mathsf{John},\ \mathsf{female\ John} = \mathsf{False}, \\ \mathsf{female\ Alice} = \mathsf{True} \end{array} \right\},$$

giving $\triangle F = \{\mathsf{female\ father} = \mathsf{False}\ !\}$. Now $(F, \mathsf{female\ father\ John} = \mathsf{True})$ is a non-theorem and has no non-extraneous side conditions apart from the conclusion itself. Thus we have no information about the truth of the conclusion from $F$. We then turn to the possible theorem

$$(F \cup \triangle F, \mathsf{female\ father\ John} = \mathsf{True}),$$

finding that it is actually impossible. Normally in a dialogue we would then answer 'No', as in Example 4.4, but since it is the probable theorem that is impossible we instead answer 'No, I don't think so'. This indicates the dependence of the answer on the conjectured beliefs.

If we instead had $F' = F \cup (\mathsf{father\ John} = \mathsf{Bob})$, the possible theorem $(F', \mathsf{female\ father\ John} = \mathsf{True})$ would have the side condition $\mathsf{female\ Bob} = \mathsf{True}$. The answer is then more complicated because we are certain that the theorem is true if $\mathsf{female\ Bob} = \mathsf{True}$ but at the same time we conjecture that it is impossible. A diplomatic response may thus be 'No, I don't think so, though it would be so if Bob was female.' $\diamond$

CHAPTER 7

# Conclusion

## 7.1. Dialogue with a Scientist

We conclude our work by revisiting the dialogues introduced in Section 4.2.1. Now we can extend the deductive reasoning used in those dialogues by the inductive procedures presented in the previous two chapters. We call the reasoner thus empowered a *scientist*.

Each time the scientist's (deductive) belief set is enlarged or revised we apply IND to the corresponding canonical rewrite system (which needs to be generated in establishing consistency). From the resulting conjectures we form an additional set of *conjectured beliefs* as either the conjectures themselves, if they are consistent with the original belief set, or a maximally consistent subset of the conjectures. The choice of the maximal subset is made by the falsifiability criterion of Definition 6.10.

For compactness, at each stage of the dialogue we only give newly declared beliefs, rather than repeating the whole belief basis. Whenever the belief basis changes we additionally give the resulting set of conjectured beliefs, although in practice this need only be generated when a question is asked of the reasoner.

EXAMPLE 7.1 (Simple answers). The first dialogue gives a simple illustration of the growing knowledge of the reasoner, both deductively and inductively, using questioning to see what it believes.

When asked a question there are now five simple possibilities (that is, those not involving side conditions) for the answer that can be returned. Let $\Gamma$ be the declared belief basis, $C$ the set of conjectured beliefs obtained from $\Gamma$, and let $f$ be the equation or inequation representing the question. Then the reasoner may answer

- 'Yes', if the question is a deductive consequence of the declared belief set, i.e. $\Gamma \Rightarrow f$;
- 'No', if the question is not a deductive consequence of the declared belief set and the corresponding theorem is *impossible*, i.e. $\Gamma \Rightarrow \neg f$;
- 'Yes, I think so', if the question is not a deductive consequence of the declared belief set but is a consequence of the belief set combined with the conjectured beliefs, i.e. $\Gamma \not\Rightarrow f$ and $\Gamma \cup C \Rightarrow f$;

- 'No, I don't think so', if the question is not a deductive consequence of the declared belief set and the theorem with the belief basis and the conjectured beliefs as hypotheses is impossible, i.e. $\Gamma \not\Rightarrow f$ and $\Gamma \cup C \Rightarrow \neg f$;
- 'I don't know', if the question is not a deductive consequence of the declared belief set and the only non-extraneous side condition is the original question itself.

The dialogue in Table 7.1 involves all five of these simple answers. ◇

EXAMPLE 7.2 (Crucial experiments). The dialogue in Tables 7.2 and 7.3 involves competing theories in the reasoner's conjectured beliefs. Now when asked a question, the simple answers which rely on conjectures are augmented by requests for the results of any crucial experiments.

The inconsistency arises when told 'The not of False is True'. Table 7.1 shows the two competing theories and their predictions. The theories disagree on the prediction for **female Alice** and so it is suggested as a crucial experiment. Meanwhile the strongest theory is taken as the working set of conjectured beliefs.

We also include in this example an illustration of the closed-minded approach to belief revision. The response to the contradictory 'John is Paul's father' is simply 'I can't believe that', quite literally indicating that the statement cannot be added to the current belief set. ◇

## 7.2. Concluding Remarks

In this work we have presented a broad system of rewrite based reasoning, through primary algorithms for deduction and induction.

Central to our deductive process was proof by contradiction, and the use of it in an equational framework motivated the extension of the standard Knuth-Bendix algorithm to a procedure for systems involving inequations. The culmination of this was a uniqueness theorem for the result of applying the algorithm to equivalent systems of equations. This then lead to an alternative proof method based on invariance.

We also presented the notion of a side condition to give an interpretation to a failed proof by contradiction, and saw it also suggested how to solve equations in a variable-free algebra. The side conditions additionally had an application as abductive hypotheses, but we have left to future work the task of using these as part of a reasoning system.

We introduced an induction algorithm for generating inductive theorems and conjectures from a database of rewrite rules. By assuming a finite database

| Human | Reasoner | Belief Set |
|---|---|---|
| John is Alice's father. | Okay. | father Alice = John<br>father $\simeq$ John ! |
| Is John the father of Alice? | Yes. | |
| Is John the father of George? | Yes, I think so. | |
| George is the father of Mary. | Okay. | father Mary = George |
| Is John the father of George? | I don't know. | |
| Is John the father of Mary? | No. | |
| John is male. | Okay. | male John = True<br>male father $\simeq$ True !<br>male $\simeq$ True ! |
| Is John male? | Yes. | |
| Is George male? | Yes, I think so. | |
| Is Alice male? | Yes, I think so. | |
| Mary is not male. | Okay. | male Mary = False<br>male father $\simeq$ True ! |
| Is Alice male? | I don't know. | |
| John is not female. | Okay. | female John = False<br>male father $\simeq$ True !<br>female father $\simeq$ False !<br>female $\simeq$ False ! |
| Alice is female. | Okay. | female Alice = True<br>male father $\simeq$ True !<br>female father $\simeq$ False ! |
| Is the father of John female? | No, I don't think so. | |

TABLE 7.1. Dialogue with simple answers

we showed that the result of this algorithm would also be finite. By further assuming that the language of the database came from a finite signature, we were able to quantify measures of belief both in isolated conjectures and in systems of conjectures. These latter systems, inductive theories, were then examined in terms of inductive belief change.

| Human | Reasoner | Belief Set |
|---|---|---|
| Peter is male. | Okay. | male Peter = True<br>male $\simeq$ True ! |
| Peter is Paul's father. | Okay. | father Paul = Peter<br>male father $\simeq$ True !<br>male $\simeq$ True ! |
| Is Alice's father male? | Yes, I think so. | |
| John is Paul's father. | I can't believe that. | |
| John is Alice's father. | Okay. | father Alice = John<br>male father $\simeq$ True !<br>male $\simeq$ True ! |
| Is John male? | Yes, I think so. | |
| Peter is not female. | Okay. | female Peter = False<br>male father $\simeq$ True !<br>male $\simeq$ True !<br>female father $\simeq$ False !<br>female $\simeq$ False ! |
| Alice is not male. | Okay. | male Alice = False<br>male father $\simeq$ True !<br>female father $\simeq$ False !<br>female $\simeq$ False ! |
| Is Paul male? | I don't know. | |
| Is Paul female? | No, I don't think so. | |

TABLE 7.2. Dialogue with crucial experiments

To achieve our broad view of reasoning we have usually restricted our attention to a simple language involving only linear terms, terms without products and their associated $n$-ary functions. However, except where noted, the algorithms and theorems presented have been given for the general term language of Section 2.1. In future work we will look at using the power of this richer language to approach larger applications present in the literature.

| Human | Reasoner | Belief Set |
|---|---|---|
| The not of True is False. | Okay. | not True = False<br>male father ≃ True !<br>female father ≃ False !<br>female ≃ False !<br>not male ≃ female<br>not male ≃ False !<br>not ≃ False ! |
| The not of False is True. | Okay. | not False = True<br>male father ≃ True !<br>female father ≃ False !<br>female ≃ False !<br>not female ≃ True !<br>not not = i |
| Is Paul male? | I don't know. | |
| Is Paul female? | No, I don't think so, but it would be useful to know if Alice is female. | |
| Alice is female. | Okay. | female Alice = False<br>not male ≃ female<br>not female ≃ male<br>female father ≃ False !<br>male father ≃ True !<br>not not = i |
| Is Paul female? | I don't know. | |

TABLE 7.3. Dialogue with crucial experiments (continued)

| Conjectures | Predictions | Falsifiability |
|---|---|---|
| not not = i | female Alice = False | 0.9375 |
| female $\simeq$ False ! | female John = False | |
| not female $\simeq$ True ! | male John = True | |
| female father $\simeq$ False ! | female Paul = False | |
| male father $\simeq$ True ! | | |
| not not = i | female Alice = True | 0.875 |
| not male $\simeq$ female | female John = False | |
| not female $\simeq$ male | male John = True | |
| female father $\simeq$ False ! | | |
| male father $\simeq$ True ! | | |

TABLE 7.4. Competing theories from the dialogue in Tables 7.2 and 7.3

# Bibliography


1. A. H. Beiler, *Recreations in the Theory of Numbers*, Dover, New York, 1966.
2. G. Le Blanc, *BMWk Revisited: Generalization and Formalization of an Algorithm for Detecting Recursive Relations in Term Sequences*, Machine Learning: ECML-94 (F. Bergadano and L. De Raedt, eds.), LNAI 784, Springer-Verlag, 1994, pp. 183–197.
3. I. Bratko, *Prolog Programming for Artificial Intelligence*, Addison-Wesley, 1990.
4. M. Bulmer, D. Fearnley-Sander, and T. Stokes, *Towards a Calculus of Algorithms*, Bulletin of the Australian Mathematical Society **50** (1994), no. 1, 81–89.
5. ———, *The Kinds of Truth of Geometry Theorems*, Submitted for publication, Dec 1995.
6. J. D. Carney, *Introduction to Symbolic Logic*, Prentice-Hall, New Jersey, 1970.
7. P. Le Chenadec, *Canonical forms in finitely presented algebras*, Pitman, London, 1986.
8. L. J. Cohen, *An Introduction to the Philosophy of Induction and Probability*, Oxford University Press, 1989.
9. D. Cox, J. Little, and D. O'Shea, *Ideals, Varieties, and Algorithms*, Springer-Verlag, 1992.
10. P. T. Cox and T. Pietrzykowski, *Causes for Events: Their Computation and Applications*, Eigth Conference on Automated Deduction (J. H. Siekmann, ed.), LNCS 230, Springer-Verlag, 1986, pp. 608–621.
11. J. Cussens, *Bayes and Pseudo-Bayes Estimates of Conditional Probabilities and their Reliability*, Machine Learning: ECML-93 (P. B. Brazdil, ed.), LNAI 667, Springer-Verlag, 1993, pp. 136–152.
12. M. Dauchet and S. Tison, *Decidability of Confluence for Ground Term Rewriting Systems*, Fundamentals of Computation Theory, FCT '85 (L. Budach, ed.), LNCS 199, Springer-Verlag, 1985, pp. 80–89.
13. N. Dershowitz, *Completion and its Applications*, Resolution of Equations in Algebraic Structures (H. Aït Kaci and M. Nivat, eds.), vol. 2, Academic Press, London, 1989, pp. 31–85.
14. N. Dershowitz, L. Marcus, and A. Tarlecki, *Existence, Uniqueness, and Construction of Rewrite Systems*, SIAM Journal of Computing **17** (1988), no. 4, 629–639.
15. D. Fearnley-Sander, *Definitional Reasoning*, Unpublished work, 1991.
16. ———, *Definitional Reasoning and Optimization*, International Workshop on Analysis and its Applications (C. V. Stanojevic and O. Hadzic, eds.), Institute of Mathematics, Novi Sad, 1992, pp. 365–379.
17. ———, *A logic of definitional reasoning*, Australian Computer Science Communications **18** (1996), 81–89.
18. W. Feller, *An Introduction to Probability Theory and its Applications*, vol. 1, John Wiley and Sons, 1968.
19. J. E. Freund, *Mathematical Statistics*, Prentice-Hall, Englewood Cliffs, NJ, 1971.
20. P. Gärdenfors, *Knowledge in Flux: Modelling the Dynamics of Epistemic States*, MIT Press, 1988.
21. M. Genesereth and N. Nilsson, *Logical Foundations of Artificial Intelligence*, Morgan Kaufmann, Palo Alto, 1988.

22. J. D. Gibson, *Principles of Digital and Analog Communications*, Macmillan, Singapore, 1990.
23. J. A. Goguen, *How to prove algebraic inductive hypotheses without induction with applications to the correctness of data type implementation*, Fifth Conference on Automated Deduction (W. Bibel and R. Kowalski, eds.), LNCS 87, Springer-Verlag, 1980, pp. 356–373.
24. S. O. Hansson, *A dyadic representation of belief*, Belief Revision (P. Gärdenfors, ed.), Cambridge University Press, Cambridge, 1992, pp. 89–121.
25. G. Hinton, *Learning Distributed Representations of Concepts*, Program of the Eight Annual Conference of the Cognitive Science Society (L. Erlbaum, ed.), Amhearst, MA, 1986.
26. J. Hsiang, H. Kirchner, P. Lescanne, and M. Rusinowitch, *The Term Rewriting Approach to Automated Theorem Proving*, Journal of Logic Programming **14** (1992), 71–99.
27. G. Huet, *A complete proof of correctness of the Knuth-Bendix completion algorithm*, Journal of Computer and System Science **23** (1981), no. 1, 11–21.
28. ———, *Cartesian Closed Categories and Lambda-Calculus*, Logical Foundations of Functional Programming (G. Huet, ed.), Addison-Wesley, Reading, Mass., 1990, pp. 7–23.
29. G. Huet and D. C. Oppen, *Equations and Rewrite Rules: A Survey*, Formal Language Theory: Perspectives and Open Problems (R. Book, ed.), Academic Press, 1980, pp. 349–405.
30. G. E. Hughes and M. J. Cresswell, *An Introduction to Modal Logic*, Methuen, London, 1968.
31. D. Kapur and D. R. Musser, *Proof by Consistency*, Artificial Intelligence **31** (1987), 125–157.
32. D. E. Knuth and P. B. Bendix, *Simple Word Problems in Universal Algebras*, Computational Problems in Abstract Algebra (J. Leech, ed.), Pergamon Press, 1970, pp. 263–297.
33. J. W. Lloyd, *Foundations of Logic Programming*, Springer-Verlag, 1987.
34. T. Mora, *Gröbner Bases for Non-Commutative Polynomial Rings*, Algebraic Algorithms and Error-Correcting Codes, AAECC-3 (J. Calmet, ed.), LNCS 229, Springer-Verlag, 1985, pp. 353–362.
35. S. Muggleton and L. De Raedt, *Inductive Logic Programming: Theory and Methods*, Journal of Logic Programming **19** (1994), 629–679.
36. J. G. Oxley, *Matroid Theory*, Oxford University Press, Oxford, 1992.
37. G. Paul, *Approaches to Abductive Reasoning: an Overview*, Artificial Intelligence Review **7** (1993), 109–152.
38. K. Popper, *Conjectures and Refutations - The Growth of Scientific Knowledge*, Routledge and Kegan Paul, London, 1974.
39. ———, *The Logic of Scientific Discovery*, Hutchinson, London, 1974.
40. W. V. Quine, *On Cores and Prime Implicants of Truth Functions*, American Mathematical Monthly **66** (1959), 755–760.
41. J. R. Quinlan, *Learning Logical Definitions from Relations*, Machine Learning **5** (1990), 239–266.
42. J. R. Quinlan and R. M. Cameron-Jones, *Living in a Closed World*, Draft paper, 1994.
43. E. Rich, *Artificial Intelligence*, McGraw-Hill, Singapore, 1986.
44. M. Ryan and M. Sadler, *Valuation Systems and Confluence Relations*, Handbook of Logic in Computer Science (S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, eds.), vol. 1, Clarendon Press, Oxford, 1992, pp. 1–78.
45. A .Sattar and A. Ghose, *Experiments in Belief Revision*, Second Automated Reasoning Day (A. Sattar, ed.), 1994, Griffith University Research Report CIT-94-20.
46. J. F. Young, *Information Theory*, Butterworth, London, 1971.

# Index