RESOLUTION THEOREM PROVING REVISITED

by

Rob S. Ballantyne

B.Sc. Simon Fraser University, 1988

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in the Department of Mathematics and Statistics

of

Simon Fraser University

SIMON FRASER UNIVERSITY

October 1995

ISBN 0-612-16783-6

APPROVAL

Name:	Rob S. Ballantyne
Degree:	Master of Science
Title of Thesis:	Resolution Theorem Proving Revisited

Examining Committee: Dr. C. Schwarz
Chair

Dr. S. K. Thomason, Senior Supervisor

Dr. A. H. Lachlan

Dr. A. R. Freedman

Dr. A. Gupta, External Examiner

Date Approved: October 26, 1995

Abstract

In 1965 Robinson published the original paper on the resolution theorem proving method. The conventions utilized in that paper would not be recognized as standard logic conventions today. For example, Robinson's notion of a model was an infinite collection of atomic or negated atomic formulas that was complete. We modernize Robinson's work by phrasing it in the context of a modern logician. Robinson, in the 1965 paper, commented that resolution calculates a proof set without making that calculation explicit; we show how these calculations can be made explicit.

Dedication

This thesis is dedicated to two individuals who due to their untimely deaths were unable to witness the conclusion of my masters degree. To my father, Paul Stewart Ballantyne, and Professor Alan Mekler; they are sorely missed.

Acknowledgments

I would like to thank Professor Steve Thomason for his encouragement, support, and patience during the completion of this thesis. Also I would like to thank my wife, Sandy Ballantyne, for her patience and understanding during the completion of the thesis.

In addition I would like to acknowledge Professor Alistair Lachlan for his frequent wise advice and the Department of Mathematics and Statistics for its support and patience.

Contents

Chapter 1

Introduction

Automating the process of formal reasoning has been a long-standing goal of logic. The search for an algorithm to determine provability or validity of first-order sentences began shortly after the description of first-order systems.

Kurt Gödel demonstrated, via the Undecidability Theorem [Godel34], that in general it is impossible for an automated procedure to determine whether a formal sentence was provable or not. This dashed the hope of reducing first-order reasoning to an algorithm and it was some years after Gödel's result before work continued on automated theorem proving. It was still, after all, possible to have an algorithm that would for any sentence establish its validity if the sentence was valid.

The first attempts at automated theorem proving were with propositional logic. Here algorithms for deciding a sentence's validity have been known for some time. The first such algorithms most likely considered truth tables for a particular sentence. If the truth table demonstrated that the sentence was true in all possible cases then that sentence was valid.

In later work Davis & Putnam [Davis60] produced an efficient method for testing propositional sentences in conjunctive normal form for satisfiability. Given a procedure that tests for satisfiability it is easy to determine the validity of a sentence. Simply apply the satisfiability test to the negation of that sentence and the original sentence is valid if and only if its negation is unsatisfiable.

The initial attempts at first-order logic basically reduced the first-order case to performing many iterations of a propositional algorithm. Each iteration of these propositional algorithms attempted to show the unsatisfiability of a sentence on a particular finite first-order structure.

Dag Prawitz [Prawitz60] first demonstrated that it is possible to combine together the propositional test for satisfiability with calculations of the potential models to produce an algorithm that didn't look at every possible model but only examined models that could help demonstrate the unsatisfiability of the sentence in question. Unfortunately Prawitz's method required that sentences be continually converted and re-converted into a particular normal form, a procedure that is very inefficient.

In [Robinson65] Robinson fused together Prawitz's idea of "calculating as you go" with Davis & Putnam's efficient method of working with conjunctive normal form sentences to produce a method called *resolution.* Resolution overcame the 're-normalizing' weakness of Prawitz's method. Sentences are converted into a normal form just once and never needed to be re-converted again.

In this thesis we revisit Robinson's paper on resolution and we update his results into the language of the modern logician, predominantly by the use of first-order structures in place of the sets of atomic and negated atomic sentence that Robinson used for models. Our approach is not unknown in the literature; Loveland [Loveland79] and Fitting [Fitting94] for example employ first-order structures. In Robinson's paper completeness of resolution follows from the completeness of ground resolution. Here that is the case as well, but we demonstrate the completeness of ground resolution with respect to first-order structures instead of Robinson's models. Other work has taken alternative approachs. Loveland [Loveland79] uses his g-models, which are similar to Robinson's models, to demonstrate the completeness of ground resolution. Fitting [Fitting94] proves completeness of resolution directly using a Model Existence Theorem. In place of the usual unification algorithm we provide a unification algorithm that is expressed as a recursive definition and is particularly succinct. We also attempt to justify a comment that Robinson left unjustified. Namely, in [Robinson65] Robinson mentions that resolution calculates a proof set without making that calculation explicit. We show how the calculation can be made explicit.

Chapter 2

Preliminaries

2.1 Language

The notion of language that we will use throughout this thesis is a slight variant of that defined by Shoenfield [Shoenfield67]. Our languages, unlike Shoenfield's, need not include an equality symbol. A *first-order* language has the following components:

1. variables: $x, y, z, w, x_1, y_1, z_1, w_1, x_2, \ldots,$
2. for each $n \geq 0$, n-ary predicate symbols and n-ary function symbols,
3. the logical symbols $\neg, \vee, \exists$.

Below we use L to denote this kind of language. The predicate and function symbols of L are called the ***non-logical symbols*** of L; these together with the variables and logical symbols constitute the ***symbols*** of L.

With respect to formulas of our language $\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{w}$ are syntactic variables that range over the variables of the language L; $\mathbf{p}, \mathbf{q}$ are syntactic variables that range over the predicate symbols; $\mathbf{f}, \mathbf{g}$ are syntactic variables that range over the function symbols; $\mathbf{a}, \mathbf{b}$ are syntactic variables that range over constant symbols (where a constant is, as usual, a 0-ary function); and $\mathbf{l}, \mathbf{m}, \mathbf{n}$, are syntactic variables that range over the non-logical (function and predicate) symbols.

Definition 2.1.1 $\mathrm{Expr}(L)$ *is the smallest set of sequences of the symbols of L closed under the following conditions:*

1. *each sequence of length one whose member is a variable of L is in* $\mathrm{Expr}(L)$,
2. *if* $\mathbf{u}_1, \ldots, \mathbf{u}_n \in \mathrm{Expr}(L)$ *and* $\mathbf{n}$ *is an n-ary non-logical symbol then* $\mathbf{n}\mathbf{u}_1 \ldots \mathbf{u}_n$ *is an element of* $\mathrm{Expr}(L)$.

$\mathrm{Term}(L)$ *is the set of members of* $\mathrm{Expr}(L)$ *which have no occurrences of predicate symbols.* $\mathrm{Atomic}(L)$ *is the set of expressions of the form* $\mathbf{p}\mathbf{a}_1 \ldots \mathbf{a}_n$ *where* $\mathbf{p}$ *is an n-ary predicate symbol and* $\mathbf{a}_1, \ldots, \mathbf{a}_n \in \mathrm{Term}(L)$. $\mathrm{Fla}(L)$ *is the least subset of* $\mathrm{Expr}(L)$ *closed under the following conditions:*

1. $\mathrm{Atomic}(L) \subseteq \mathrm{Fla}(L)$,
2. *if* $\mathbf{A} \in \mathrm{Fla}(L)$ *then* $\neg\mathbf{A} \in \mathrm{Fla}(L)$,
3. *if* $\mathbf{A}, \mathbf{B} \in \mathrm{Fla}(L)$ *then* $\vee\mathbf{A}\mathbf{B} \in \mathrm{Fla}(L)$,
4. *if* $\mathbf{A} \in \mathrm{Fla}(L)$ *then* $\exists\mathbf{x}\mathbf{A} \in \mathrm{Fla}(L)$.

An element of $\mathrm{Expr}(L)$ is called an *expression* of the language and an element of $\mathrm{Term}(L)$ is called a *term* of the language. A member of $\mathrm{Atomic}(L)$ is an *atomic formula*, and an element of $\mathrm{Fla}(L)$ a *formula*. We drop the reference L, to the language, and write Expr, Term, Atomic, and Fla for Expr(L), Term(L), Atomic(L), and Fla(L) respectively, when it is clear which language is under discussion.

We define the *free variables* of an expression inductively as follows. The free variables of a term are the variables occurring in that term. The free variables: in $\neg\mathbf{A}$ are the free variables in $\mathbf{A}$, in $\wedge\mathbf{AB}$ are the free variables in either $\mathbf{A}$ or $\mathbf{B}$, in $\exists\mathbf{xA}$ are the free variables in $\mathbf{A}$ except $\mathbf{x}$. An expression is a *sentence* if it has no free variables.

A formula which is either atomic or negated atomic is called a *literal.* For any literal $\mathbf{A}$ we define the *complement* $\mathbf{A}'$ of $\mathbf{A}$ as:

$$\mathbf{A}' = \begin{cases} \neg\mathbf{A} & \text{if } \mathbf{A} \text{ is atomic} \\ \mathbf{B} & \text{if } \mathbf{A} = \neg\mathbf{B}. \end{cases}$$

Notice that the complement of a literal is a literal and not a more complicated formula.

At this point we admit the use of the usual logical connectives and quantifiers as abbreviations. We also admit the usual infix notation for connectives, as an "abbreviation."

2.2 Structures

We now define the notion of truth, that is, the assignment of truth values to formulas. We follow the presentation given by Enderton in [Enderton72].

Definition 2.2.1 *A* structure $\mathfrak{A}$ *for a language* L *consists of: a non-empty set* $|\mathfrak{A}|$ *called the* universe, *for each* n*-ary function symbol* $\mathbf{f}$ *of* L *a corresponding* n*-ary function* $\mathbf{f}_{\mathfrak{A}} : |\mathfrak{A}|^n \longrightarrow |\mathfrak{A}|$, *and for each* n*-ary predicate symbol* $\mathbf{p}$ *of* L *an* n*-ary relation*

$p_{\mathfrak{A}} \subseteq |\mathfrak{A}|^n$. *A* sub-structure *of a* $\mathfrak{A}$ *is a structure* $\mathfrak{A}'$ *such that* $|\mathfrak{A}'| \subseteq |\mathfrak{A}|$, *the predicates* $\mathbf{p}_{\mathfrak{A}'}$ *of* $\mathfrak{A}'$ *are restrictions of the corresponding predicates* $\mathbf{p}_{\mathfrak{A}}$ *of* $\mathfrak{A}$ *to* $|\mathfrak{A}'|$, *and the functions* $\mathbf{f}_{\mathfrak{A}'}$ *of* $\mathfrak{A}'$ *are restrictions of the corresponding functions* $\mathbf{f}_{\mathfrak{A}}$ *of* $\mathfrak{A}$ *to* $|\mathfrak{A}'|$.

Definition 2.2.2 *A* valuation *for a structure* $\mathfrak{A}$ *is a function that maps the set of variables of the language to* $|\mathfrak{A}|$.

We can inductively extend any valuation s to a function $\bar{s} : \text{Term} \to |\mathfrak{A}|$ in the following way. Let $\mathbf{t} \in \text{Term}$; then either $\mathbf{t}$ is a variable or $\mathbf{t}$ is $\mathbf{f}\mathbf{u}_1 \ldots \mathbf{u}_n$. If $\mathbf{t}$ is a variable then set $\bar{s}(\mathbf{t}) = s(\mathbf{t})$, otherwise set $\bar{s}(\mathbf{t}) = \mathbf{f}_{\mathfrak{A}}(\bar{s}(\mathbf{u}_1), \ldots, \bar{s}(\mathbf{u}_n))$. For any valuation s and any $a \in \mathfrak{A}$, and variable $\mathbf{x}$ the valuation $s(\mathbf{x}/a)$ is defined by

$$s(\mathbf{x}/a)(\mathbf{y}) = \begin{cases} a & \text{if } \mathbf{y} = \mathbf{x} \\ s(\mathbf{y}) & \text{otherwise.} \end{cases}$$

Definition 2.2.3 *For* $\mathbf{A} \in \text{Fla}(L)$, $\mathfrak{A}$ *a structure for* L, *and* s *a valuation for* $\mathfrak{A}$, *we say that* $\mathfrak{A}$ *makes* $\mathbf{A}$ true with s *(and we write* $\models_{\mathfrak{A}} \mathbf{A}[s]$ *) if*

1. $\mathbf{A}$ *is* $\mathbf{p}\mathbf{t}_1 \ldots \mathbf{t}_n$ *and* $(\bar{s}(\mathbf{t}_1), \ldots, \bar{s}(\mathbf{t}_n)) \in \mathbf{p}_{\mathfrak{A}}$, *or*

2. $\mathbf{A}$ *is* $\neg\mathbf{B}$ *and it is not the case that* $\models_{\mathfrak{A}} \mathbf{B}[s]$, *or*

3. $\mathbf{A}$ *is* $\vee\mathbf{A}\mathbf{B}$ *and either* $\models_{\mathfrak{A}} \mathbf{A}[s]$ *or* $\models_{\mathfrak{A}} \mathbf{B}[s]$, *or*

4. $\mathbf{A}$ *is* $\exists\mathbf{x}\mathbf{B}$ *and there exists* $a \in |\mathfrak{A}|$ *such that* $\models_{\mathfrak{A}} \mathbf{B}[s(\mathbf{x}/a)]$.

If $\mathbf{A}$ is a sentence in the language L then a structure $\mathfrak{A}$, for the language L, will make that sentence true or false independent of the valuation s and we write $\models_{\mathfrak{A}} \mathbf{A}$ instead of $\models_{\mathfrak{A}} \mathbf{A}[s]$. For a sentence $\mathbf{A}$ if $\models_{\mathfrak{A}} \mathbf{A}$ we say that $\mathfrak{A}$ *models* or *satisfies* $\mathbf{A}$. We say that a sentence $\mathbf{A}$ is *valid* if every structure for the language L satisfies that sentence, and if no such structure satisfies $\mathbf{A}$ then we say that $\mathbf{A}$ is *unsatisfiable.*

2.3 The Clausal Form of Sentences

A sentence $\mathbf{A}$ is said to be *prenex* if it is of the form $Q_1\mathbf{x}_1 \ldots Q_n\mathbf{x}_n\mathbf{B}$ where each Q_i is either $\exists$ or $\forall$ and $\mathbf{B}$ is quantifier-free; in this case $\mathbf{B}$ is referred to as the *matrix* of $\mathbf{A}$. If a sentence is prenex and all of its quantifiers are universal then the sentence is said to be *universal*. A formula $\mathbf{A}$ is in *conjunctive normal form* if $\mathbf{A}$ is of the form $\mathbf{B}_1 \wedge \mathbf{B}_2 \wedge \ldots \wedge \mathbf{B}_n$ and each of the $\mathbf{B}_i$ is a disjunction of literals.

The purpose of this section is to demonstrate that for any sentence $\mathbf{A}$ we can find a universal sentence $\mathbf{B}$ with its matrix in conjunctive normal form and which is satisfiable if and only if $\mathbf{A}$ is satisfiable. The results needed to show this are standard and below we only describe the conversion process. See the appropriate sections of [Shoenfield67] for details.

We convert a formula into a prenex form formula by "pushing" the quantifiers outward, taking care to change the sense of the quantifier (change $\forall$ to $\exists$ and $\exists$ to $\forall$) if we push the quantifier past a negation symbol. We also take care not to capture variables in the scope of quantifiers that previously did not bind them by renaming the variables as required (see [Shoenfield67], page 36). In this way we obtain a formula, in prenex form, which is provably equivalent to the original formula.

We then obtain a universal formula from the prenex formula above by eliminating the existential quantifiers by the use of Skolem functions (see [Shoenfield67], page 56). At this point we have a universal formula in an enlarged language (more function symbols) such that the universal formula is satisfiable if and only if the original is.

Finally we convert the matrix of the universal formula to conjunctive normal form and thus we have obtained a formula which is provably equivalent to the universal formula and that is satisfiable if and only if the original formula was satisfiable.

Example 2.3.1 *Consider the sentence* $\neg\exists y(\forall x P(x)) \wedge (\forall x Q(x,y))$. *After moving the quantifiers outward we obtain* $\forall y \exists x_1 \exists x_2 \neg(P(x_1) \wedge Q(x_2, y))$. *The next step requires that we remove all existential quantifiers by use of Skolem functions, thus we obtain* $\forall y \neg(P(a) \wedge Q(f(a), y))$. *Now we convert the matrix of this sentence to conjunctive normal form and obtain* $\forall y(\neg P(a) \vee \neg Q(f(a), y))$.

We represent a universal prenex sentence

$$\forall \mathbf{x}_1 \forall \mathbf{x}_2 \ldots \forall \mathbf{x}_n \bigwedge_{i=1}^{k} \left(\bigvee_{j=1}^{l_i} \mathbf{A}_{i,j} \right)$$

(where the $\mathbf{A}_{i,j}$'s are literals) as a set:

$$\langle [\mathbf{A}_{1,1}, \ldots, \mathbf{A}_{1,l_1}], [\mathbf{A}_{2,1}, \ldots, \mathbf{A}_{2,l_2}], \ldots, [\mathbf{A}_{k,1}, \ldots, \mathbf{A}_{k,l_k}] \rangle .$$

where it is understood that $[\mathbf{B}_1, \ldots, \mathbf{B}_n]$ is the set $\{\mathbf{B}_1, \ldots, \mathbf{B}_n\}$ considered as a disjunction of the $\mathbf{B}_i$'s and $\langle C_1, \ldots, C_n \rangle$ is the set $\{C_1, \ldots, C_n\}$ considered as a conjunction of the C_i's.

We call a set $[\mathbf{A}_1, \ldots, \mathbf{A}_i]$, where the $\mathbf{A}_j$'s are literals, a *clause*. The empty clause is denoted $\Box$. Note that many sentences may be represented by one single set of clauses, because repetition and order of conjuncts and disjuncts is lost in the passage to the set of clauses. This is not a problem, however, because these sentences are satisfied by exactly the same structures.

If $C = [\mathbf{B}_1, \ldots, \mathbf{B}_k]$ is a clause we write $\models_{\mathfrak{A}} C$ to mean $\models_{\mathfrak{A}} \forall \mathbf{x}_1 \ldots \forall \mathbf{x}_n \bigvee_{i=1}^{k} \mathbf{B}_i$ when $\mathbf{x}_1 \ldots \mathbf{x}_n$ are all the variables that occur in the literals of C. When S is a set of clauses we write $\models_{\mathfrak{A}} S$ to mean that $\models_{\mathfrak{A}} C$ for each $C \in S$.

Chapter 3

Ground Resolution

In this chapter we present an introduction to resolution, a proof method that is especially amenable to automated theorem proving due mainly to the fact that resolution has just one inference rule (and no axioms) to implement. For the time being we consider resolution only for sets of clauses in which no variables occur, the so-called *ground clauses.*

3.1 Definitions

Definition 3.1.1 *A term with no variables is called a* ground term *and a clause composed of literals with no variables is called a* ground clause.

Definition 3.1.2 *With any set S of clauses there is associated a set* $\mathrm{Herb}(S)$ *of ground terms, called the* Herbrand universe of S. *We define* $\mathrm{Herb}(S)$ *inductively as follows: let T be the set of constants occurring in S (if S has no constants then let $T = \{\mathbf{a}\}$, where* **a** *is a constant symbol). Then* $\mathrm{Herb}(S)$ *is the least set of terms* *containing T such that if* $\mathbf{f}$ *is an n-ary function symbol occurring in S and* $\mathbf{t}_1, \ldots, \mathbf{t}_n$ *are elements of* $\mathrm{Herb}(S)$ *then* $\mathbf{f}\mathbf{t}_1 \ldots \mathbf{t}_n \in \mathrm{Herb}(S)$.

Definition 3.1.3 *If S is any set of clauses and P is any set of terms then the* saturation of S with P, *denoted $P(S)$, is the set of all clauses obtained from the clauses in S by substituting for each variable a term in P. Where P is the Herbrand universe of a set of clauses S we call any element of $P(S)$ a* Herbrand instance *of the clause of S from which it was derived.*

3.2 Resolution

Definition 3.2.1 *If C and D are ground clauses, $\mathbf{A} \in C$, and $\mathbf{A}' \in D$ then $(C - \{\mathbf{A}\}) \cup (D - \{\mathbf{A}'\})$ is a* ground resolvant *of C and D.*

Producing resolvants of two clauses is the basis of our inference rule for first-order logic so we need to show that the rule is sound.

Proposition 3.2.1 *Let S be a set of ground clauses, let $C, D \in S$, and let R be a ground resolvant of C and D. If $\models_{\mathfrak{A}} S$ then $\models_{\mathfrak{A}} S \cup \{R\}$.*

Proof: If R is a ground resolvant of C and D then $R = (C - \{\mathbf{A}\}) \cup (D - \{\mathbf{A}'\})$ for some ground literal $\mathbf{A}$. Now suppose $\models_{\mathfrak{A}} S$ and for a contradiction suppose $\mathfrak{A}$ does not model $S \cup \{R\}$. Clearly $\mathfrak{A}$ satisfies C but not $C - \{\mathbf{A}\}$, so $\mathfrak{A}$ makes $\mathbf{A}$ true. Similarly $\mathfrak{A}$ satisfies D but not $D - \{\mathbf{A}'\}$, so $\mathfrak{A}$ makes $\mathbf{A}'$ true; but this is a contradiction. ∎

Definition 3.2.2 *If S is any set of ground clauses, then the* ground resolution *of S, denoted $\mathcal{R}(S)$, is S together with all ground resolvants of all pairs of clauses in S. The n-th* ground resolution, *$\mathcal{R}^n(S)$, is defined inductively by: $\mathcal{R}^0(S) = S$ and $\mathcal{R}^{n+1}(S) = \mathcal{R}(\mathcal{R}^n(S))$.*

Theorem 3.2.1 (The Ground Resolution Theorem) *Let S be any set of ground clauses. Then S is unsatisfiable if and only if $\square \in \mathcal{R}^n(S)$ for some $n \geq 0$.*

Given the Ground Resolution theorem (the proof will be along in a moment) we now have a procedure for testing the validity of any variable-free sentence $\mathbf{A}$:

1. Write $\neg\mathbf{A}$ as a set S of ground clauses.
2. Set $n \leftarrow 1$.
3. While $\mathcal{R}^n(S) \neq \mathcal{R}^{n+1}(S)$ and $\square \notin \mathcal{R}^{n+1}(S)$
   set $n \leftarrow n + 1$.
4. If $\square \in \mathcal{R}^{n+1}(S)$ then $\neg\mathbf{A}$ is unsatisfiable and hence $\mathbf{A}$ is valid. Otherwise $\mathcal{R}^n(S) = \mathcal{R}^{n+1}(S)$ and $\mathcal{R}^{n+1}(S)$ is satisfiable and so $\neg\mathbf{A}$ is satisfiable and $\mathbf{A}$ is not valid.

Notice that if S is finite then step 3 of the procedure will not loop forever; there are only finitely many distinct literals in S and hence there are only finitely many clauses of those literals and hence not all of $\mathcal{R}^0(S), \mathcal{R}^1(S), \mathcal{R}^2(S), \ldots$ can be distinct.

Example 3.2.1 *Let $p, q,$ and r be 0-ary predicate symbols. Then the variable free sentence $(p \to q) \to ((q \to r) \to (p \to r))$ is a tautology.*

The negation of this sentence is equivalent to $(\neg p \vee q) \wedge (\neg q \vee r) \wedge p \wedge \neg r$. So let

$$S = \langle [\neg p, q], [\neg q, r], [p], [\neg r] \rangle$$

and then

$$\mathcal{R}^1(S) = S \cup \langle [\neg p, r], [q], [\neg q] \rangle$$

and

$$\mathcal{R}^2(S) = \mathcal{R}(S) \cup \langle [r], [\neg p], \Box \rangle$$

and since $\Box \in \mathcal{R}^2(S)$ we stop and answer that $R^2(S)$ is unsatisfiable and have hence refuted the negation of $(p \to q) \to ((q \to r) \to (p \to r))$ and have shown the sentence valid.

Before we can prove the Ground Resolution Theorem we require the notion of a partial truth valuation.

Definition 3.2.3 *A* partial truth valuation *for L is a function v that maps a subset of the set of all ground atomic sentences of L into $\{\mathsf{T}, \mathsf{F}\}$. A* truth valuation *is a partial truth valuation which is total on the set of ground atomic sentences of L. We can extend a partial truth valuation v to a partial function $\bar{v}$ that maps literals, clauses, and sets of clauses to $\{\mathsf{T}, \mathsf{F}\}$ in the following way. For a literal $\mathbf{A}$ we define:*

$$\bar{v}(\mathbf{A}) = \begin{cases} v(\mathbf{A}) & \textit{if } \mathbf{A} \textit{ is atomic} \\ \mathsf{T} & \textit{if } \mathbf{A} = \neg\mathbf{B} \textit{ and } v(\mathbf{B}) = \mathsf{F} \\ \mathsf{F} & \textit{if } \mathbf{A} = \neg\mathbf{B} \textit{ and } v(\mathbf{B}) = \mathsf{T} \\ \textit{undefined} & \textit{if } \mathbf{A} = \neg\mathbf{B} \textit{ and } v(\mathbf{B}) \textit{ is undefined.} \end{cases}$$

For a clause C, $\bar{v}(C) = \mathsf{T}$ unless $\bar{v}(\mathbf{A}) = \mathsf{F}$ for every literal $\mathbf{A} \in C$, in which case $\bar{v}(C) = \mathsf{F}$. Finally $\bar{v}(S) = \mathsf{T}$ for a set S of clauses if and only if $\bar{v}(C) = \mathsf{T}$ for every clause $C \in S$ and $\bar{v}(S) = \mathsf{F}$ if and only if there is a clause $C \in S$ for which $\bar{v}(C) = \mathsf{F}$.

Note that $\bar{v}(\Box) = \mathsf{F}$ and that if $\Box \in S$ then $\bar{v}(S) = \mathsf{F}$. Also note that $\bar{v}$ is a total function with respect to either clauses or sets of clauses.

Lemma 3.2.1 *If S is any finite set of ground clauses, then there is a truth valuation v such that $\bar{v}(S) = \mathsf{T}$ if and only if there is no n for which $\Box \in \mathcal{R}^n(S)$.*

Proof: ($\Rightarrow$) Suppose $\bar{v}(S) = \mathsf{T}$. By an analog of the proof of Proposition 3.2.1 it is easy to see that: if $\bar{v}(S) = \mathsf{T}$ then $\bar{v}(\mathcal{R}(S)) = \mathsf{T}$. Thus, if $\bar{v}(S) = \mathsf{T}$ then $\bar{v}(\mathcal{R}^n(S)) = \mathsf{T}$ for all $n \in \omega$, so $\Box \notin \mathcal{R}^n(S)$ for all $n \in \omega$.

($\Leftarrow$) Suppose there is no n such that $\Box \in \mathcal{R}^n(S)$. Let T be the first $\mathcal{R}^n(S)$ for which $\mathcal{R}^n(S) = \mathcal{R}^{n+1}(S)$, i.e. T is the terminating set of the resolution procedure. We proceed by constructing a sequence of partial truth valuations that ends with a valuation that assigns a truth value to all the literals in T. This partial truth valuation is then extended to a total valuation in an arbitrary way but still maps T to T.

Let $\mathbf{A}_1, \ldots, \mathbf{A}_k$ be all the distinct atomic formulas such that each $\mathbf{A}_i$ or its complement occurs in a clause of T and let v_0 be the empty partial truth valuation. From v_i we obtain v_{i+1} by first letting

$$w_{i+1}(\mathbf{B}) = \begin{cases} v_i(\mathbf{B}) & \text{if } \mathbf{B} \in \mathrm{dom}(v_i) \\ \mathsf{T} & \text{if } \mathbf{B} = \mathbf{A}_{i+1} \\ \mathit{undefined} & \text{otherwise.} \end{cases}$$

Now we take v_{i+1} to be w_{i+1} unless there is a clause $C \in T$ such that $\bar{w}_{i+1}(C) = \mathsf{F}$, in which case we let

$$v_{i+1}(\mathbf{B}) = \begin{cases} v_i(\mathbf{B}) & \text{if } \mathbf{B} \in \mathrm{dom}(v_i) \\ \mathsf{F} & \text{if } \mathbf{B} = \mathbf{A}_{i+1} \\ \mathit{undefined} & \text{otherwise.} \end{cases}$$

At the end of this procedure we set

$$v(\mathbf{A}) = \begin{cases} v_k(\mathbf{A}) & \text{if } \mathbf{A} \text{ is in the domain of } v_k \\ \mathsf{F} & \text{otherwise.} \end{cases}$$

We claim that $\bar{v}(T) = \mathsf{T}$. The proof is by induction.

Clearly, $\bar{v}_0(T) = \mathsf{T}$ because $\Box \notin T$ and v_0 is the empty partial truth valuation.

Now suppose there is a least stage, say i, where $\bar{v}_i(T) = \mathsf{F}$. Then there is a $C \in T$ with $\bar{v}_i(C) = \mathsf{F}$. Notice that $v_i \neq w_i$ because v_i is set equal to w_i only if w_i maps every element of T to T. From this we conclude that $\bar{v}_i(\mathbf{A}_i) = \mathsf{F}$. Now since $\bar{v}_{i-1}(C) = \mathsf{T}$ and $\bar{v}_i(C) = \mathsf{F}$ we can see that $\mathbf{A}_i \in C$ and $\bar{v}_{i-1}$ maps all the elements of C, except $\mathbf{A}_i$, to F. Now because v_i was not chosen to be w_i there must be a $D \in T$ such that $\bar{w}_i(D) = \mathsf{F}$ and thus $\neg\mathbf{A}_i \in D$ and $\bar{v}_{i-1}$ maps all the elements of D, except $\neg\mathbf{A}_i$, to F. So the clause $R = (C - \{\mathbf{A}_i\}) \cup (D - \{\neg\mathbf{A}_i\})$ is an element of T because T is closed under resolution. And $\bar{v}_{i-1}(\mathbf{B}) = \mathsf{F}$ for all $\mathbf{B} \in R$. So i is not the least stage where $\bar{v}_i(T) = \mathsf{F}$. This is a contradiction, thus $\bar{v}(T) = \mathsf{T}$ and v is total. ∎

Proof (of the Ground Resolution Theorem):

($\Leftarrow$) Suppose that S is satisfied by $\mathfrak{A}$; then it is clear by Proposition 3.2.1 that $\mathfrak{A}$ also models $\mathcal{R}^n(S)$, but $\not\models_{\mathfrak{A}} \Box$, so $\Box \notin \mathcal{R}^n(S)$.

($\Rightarrow$) By the foregoing lemma, let v be a truth valuation such that $\bar{v}(S) = \mathsf{T}$. Now from v we obtain a structure $\mathfrak{A}$ that satisfies S.

Let the universe of our model $\mathfrak{A}$ be Herb(S) and for each function symbol $\mathbf{f}$ in the language of S let $\mathbf{f}_{\mathfrak{A}}(\mathbf{t}_1, \ldots, \mathbf{t}_n) = \mathbf{f}\mathbf{t}_1 \ldots \mathbf{t}_n$. Finally for each predicate symbol $\mathbf{p}$ let $(\mathbf{t}_1, \ldots, \mathbf{t}_n) \in \mathbf{p}_{\mathfrak{A}}$ if and only if $v(\mathbf{p}\mathbf{t}_1 \ldots \mathbf{t}_n) = \mathsf{T}$. Notice that $\mathfrak{A}$ makes an atomic sentence true if and only if v assigns the value T to that sentence. So v and $\mathfrak{A}$

"agree" on the truth values for the atomic sentences, therefore they must agree upon all clauses in S and therefore on S itself and thus $\models_{\mathfrak{A}} S$. ∎

3.3 First-Order Theorem Proving

We are able, so far, to determine the unsatisfiability of a set of ground clauses with the ground resolution procedure. Here we show that to determine the unsatisfiability of a set of arbitrary (not necessarily ground) clauses it is sufficient to examine a set of ground clauses that are Herbrand instances of the original set of clauses and determine if that set is unsatisfiable.

Theorem 3.3.1 *Let* $\mathbf{A}$ *be a universal prenex sentence and let* S *be* $\mathbf{A}$*'s representation as a set of clauses; then* $\mathbf{A}$ *is unsatisfiable if and only if* $P(S)$ *is unsatisfiable, where* P *is the Herbrand universe of* S.

Proof: ($\Leftarrow$) If $\mathfrak{A}$ satisfies $\mathbf{A}$ then, since $\mathbf{A}$ is universal, $\mathfrak{A}$ satisfies every Herbrand instance of the matrix of $\mathbf{A}$, so $\models_{\mathfrak{A}} P(S)$.

($\Rightarrow$) Let $\mathbf{A}$ be $\forall \mathbf{x}_1 \ldots \forall \mathbf{x}_n \mathbf{B}$ where $\mathbf{B}$ is quantifier-free, and suppose that there is a model $\mathfrak{A}$ of $P(S)$. Let $\mathfrak{A}'$ be the sub-structure of $\mathfrak{A}$ to just the elements of $|\mathfrak{A}|$ that are named by the members of P. The structure $\mathfrak{A}'$ will make $\mathbf{A}$ true if $\models_{\mathfrak{A}'} \mathbf{B}[s]$ for every s mapping each of the variables occurring in $\mathbf{B}$ to the members of $|\mathfrak{A}'|$. But the elements of $|\mathfrak{A}'|$ are just those elements named by members of P so $\models_{\mathfrak{A}'} \mathbf{A}$ because $\models_{\mathfrak{A}'} P(S)$. ∎

Lemma 3.3.1 *A prenex universal sentence* $\forall \mathbf{x}_1 \ldots \forall \mathbf{x}_n \mathbf{A}$ *is unsatisfiable if and only if some finite subset of* $P(S)$ *is unsatisfiable, where* S *represents* $\forall \mathbf{x}_1 \ldots \forall \mathbf{x}_n \mathbf{A}$ *as a* *set of clauses and* $P = \text{Herb}(S)$.

Proof: From the above theorem we know that $\forall \mathbf{x}_1 \ldots \forall \mathbf{x}_n \mathbf{A}$ is unsatisfiable if and only if the set $P(S)$ is unsatisfiable, and by The Compactness Theorem the set $P(S)$ is unsatisfiable if and only if some finite subset of it is. ■

Lemma 3.3.2 *A set S of clauses is unsatisfiable if and only if there is some finite $P \subseteq \text{Herb}(S)$ such that $P(S)$ is unsatisfiable.*

Proof: ($\Leftarrow$) Let $P \subseteq \text{Herb}(S)$ be finite and suppose $P(S)$ is unsatisfiable. Now $P(S)$ is essentially a set of Herbrand instances of $\mathbf{A}$, where $\forall \mathbf{x}_1 \ldots \forall \mathbf{x}_n \mathbf{A}$ is the sentence that S represents. Since $P(S)$ is unsatisfiable, from Lemma 3.3.1, $\forall \mathbf{x}_1 \ldots \forall \mathbf{x}_n \mathbf{A}$ is unsatisfiable and hence S is unsatisfiable.
($\Rightarrow$) A set of clauses represents a universal prenex sentence $\forall \mathbf{x}_1 \ldots \forall \mathbf{x}_n \mathbf{A}$ where $\mathbf{A}$ is in conjunctive normal form. From the previous lemma we know that $\forall \mathbf{x}_1 \ldots \forall \mathbf{x}_n \mathbf{A}$ is unsatisfiable if and only if some finite set, say Γ, of Herbrand instances of $\mathbf{A}$ is unsatisfiable. Each instance of $\mathbf{A}$ is again in prenex conjunctive normal form and Γ is therefore essentially set of ground clauses. Let P comprise the elements of $\text{Herb}(S)$ that occur in Γ. Now $\Gamma \subseteq P(S)$ and $P(S)$ is unsatisfiable if Γ is. ■

Given Lemma 3.3.2 we now have a procedure to determine the satisfiability of a set S of arbitrary clauses: we must search all the finite subsets of $\text{Herb}(S)$ for a set P such that the set $P(S)$ of ground clauses is unsatisfiable. Of course we use ground resolution to check $P(S)$ for unsatisfiability.

At this point we have a procedure that will test for the satisfiability of any given sentence of first order logic, where we first write that sentence in an equivalently-satisfiable clausal form, say S. A drawback to this procedure is that we must generate a sequence $P_0 \subseteq P_1 \subseteq \ldots$ of finite subsets of Herb(S) with the property that $\bigcup_{n \in \omega} P_n =$ Herb(S) then in sequence perform ground resolution on $P_i(S)$, for $1 \leq i$. When we find a P_j for which ground resolution halts with an empty clause that P_j is called a *proof set* for S. Our procedure then is, in simple terms, to search out a proof set amongst the finite subsets of the Herbrand universe for the given clause. This search may not end and even when it does the number of potential proof sets we have to search through may be extremely large. In addition each ground resolution calulation carried out at every stage of the above procedure may require exponential amounts of time (in terms of the input size), as shown by Urquhart [Urquhart87].

Resolution (freed from the restriction to ground clauses) will not have this drawback. Resolution, it turns out, attempts to calculate the proof set of a set of clauses as it resolves upon those clauses.

Chapter 4

Substitution and Unification

While ground resolution forms resolvants from ground clauses, the general resolution procedure has to form resolvants out of clauses of literals that have occurrences of variables. In the previous chapter we noted that resolution attempts to calculate the proof set of a set of clauses. It is this part of the resolution procedure, the part that deals with the variables, that generates a proof set. This sub-procedure, called unification, is the process of finding substitutions that make two or more literals identical.

4.1 Substitutions

Definition 4.1.1 *A* substitution *θ is a function from a set of variables to a set of terms. From a substitution θ we can obtain $\bar{\theta}$, a function from a set of expressions to* *a set of expressions defined by:*

$$\mathbf{t}\bar{\theta} = \begin{cases} \mathbf{t}\theta & \textit{if } \mathbf{t} \textit{ is a variable} \\ \mathbf{l}(\mathbf{t}_1\bar{\theta})\ldots(\mathbf{t}_n\bar{\theta}) & \textit{if } \mathbf{t} \textit{ is the expression } \mathbf{l}\mathbf{t}_1\ldots\mathbf{t}_n. \end{cases}$$

Further, if $\mathbf{t}$ *is a literal we define* $(\neg\mathbf{t})\bar{\theta}$ *to be* $\neg(\mathbf{t}\bar{\theta})$.

The need to apply a substitution to each element of a clause or to each element of every member of a set of clauses arises. We write $C\bar{\theta}$ and $S\bar{\theta}$ for the sets $\{\mathbf{t}\bar{\theta} : \mathbf{t} \in C\}$ and $\{D\bar{\theta} : D \in S\}$ respectively. From now on we write θ instead of $\bar{\theta}$; no confusion should arise from this shorthand.

If θ is a substitution such that $\mathbf{x}\theta \neq \mathbf{x}$ for only finitely many variables $\mathbf{x}$ we represent θ by the set $\{\mathbf{x} \mapsto \mathbf{x}\theta : \mathbf{x}\theta \neq \mathbf{x}\}$. A set $\{\mathbf{x}_1 \mapsto \mathbf{t}_1, \ldots, \mathbf{x}_n \mapsto \mathbf{t}_n\}$, where $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are distinct, defines a substitution θ via:

$$\mathbf{z}\theta = \begin{cases} \mathbf{t}_i & \text{if } \mathbf{z} = \mathbf{x}_i \\ \mathbf{z} & \text{otherwise.} \end{cases}$$

From this point forward we no longer make a distinction between a representation of a substitution and the substitution itself; we write $\theta = \{\mathbf{x}_1 \mapsto \mathbf{t}_1, \ldots, \mathbf{x}_n \mapsto \mathbf{t}_n\}$ and refer to this set as a substitution.

It is clear that the substitution represented by $\{\}$ is the identity function. If C is a clause and $\mathbf{z}_1, \ldots, \mathbf{z}_n$ are all the distinct variables that occur in C (in some fixed order) then the substitutions

$$\xi_C = \{\mathbf{z}_1 \mapsto x_1, \ldots, \mathbf{z}_n \mapsto x_n\}$$

and $\eta_C = \{\mathbf{z}_1 \mapsto y_1, \ldots, \mathbf{z}_n \mapsto y_n\}$ are called the x-standardization and the y-standardization of C respectively.

Lemma 4.1.1 *Let* $\theta = \{\mathbf{x}_1 \mapsto \mathbf{t}_1, \ldots, \mathbf{x}_n \mapsto \mathbf{t}_n\}$ *and* $\sigma = \{\mathbf{y}_1 \mapsto \mathbf{u}_1, \ldots, \mathbf{y}_m \mapsto \mathbf{u}_m\}$. *Then* $\theta \circ \sigma = \{\mathbf{y}_i \mapsto \mathbf{u}_i\theta : 1 \leq i \leq m\} \cup \{\mathbf{x}_i \mapsto \mathbf{t}_i : 1 \leq i \leq n \wedge (\forall j)_{1 \leq j \leq m}(\mathbf{x}_i \neq \mathbf{y}_j)\}$.

Proof:

Let $\lambda = \{\mathbf{y}_i \mapsto \mathbf{u}_i\theta : 1 \leq i \leq m\} \cup \{\mathbf{x}_i \mapsto \mathbf{t}_i : 1 \leq i \leq n \wedge (\forall j)_{1 \leq j \leq m}(\mathbf{x}_i \neq \mathbf{y}_j)\}$. It is sufficient to see that $\theta \circ \sigma$ and λ assign the same value to each of the variables $\mathbf{x}_1, \ldots, \mathbf{x}_n, \mathbf{y}_1, \ldots, \mathbf{y}_m$. Let $\mathbf{y}_j$ be an arbitrary one of the $\mathbf{y}$'s. Then $\mathbf{y}_j\lambda = \mathbf{u}_j\theta$ and $\mathbf{y}_j(\theta \circ \sigma) = (\mathbf{y}_j\sigma)\theta = \mathbf{u}_j\theta$. Now let $\mathbf{x}_i$ be an arbitrary one of the $\mathbf{x}$'s that is not a $\mathbf{y}$; then $\mathbf{x}_i\lambda = \mathbf{t}_i$ and $(\mathbf{x}_i\sigma)\theta = \mathbf{x}_i\theta = \mathbf{t}_i$. ∎

Definition 4.1.2 *Let A be a set of atomic formulas or terms and let θ be a substitution such that* $\mathbf{t}\theta = \mathbf{u}\theta$ *for all* $\mathbf{t}, \mathbf{u} \in A$*; then θ is a* unifier *of the set A. Further if θ is a unifier of A such that for any unifier λ of A there exists a substitution σ such that $\lambda = \sigma \circ \theta$ then θ is a* most general unifier *of A.*

We require the notion of a unifier for a pair of sequences. If $\langle \mathbf{t}_1, \ldots, \mathbf{t}_n \rangle$ and $\langle \mathbf{u}_1, \ldots, \mathbf{u}_n \rangle$ are two sequences and θ is a substitution such that $\mathbf{t}_i\theta = \mathbf{u}_i\theta$ for each $1 \leq i \leq n$ we say that θ unifies the two sequences. As in Definition 4.1.2, if θ is a unifier of a pair of sequences such that for any unifier λ of those two sequences $\lambda = \sigma \circ \theta$ then θ is called a most general unifier of that pair of sequences.

4.2 Unification

Unification is the process of finding a most general unifier for a set of formulas. We proceed by showing that unifying pairs of expressions is sufficient, since a unifier of a set can be calculated by considering the elements pairwise. Finally we provide a unification algorithm for pairs of expressions and prove it correct.

Lemma 4.2.1 *Let $\langle \mathbf{t}_1, \ldots, \mathbf{t}_k \rangle$ and $\langle \mathbf{u}_1, \ldots, \mathbf{u}_k \rangle$ be two sequences most generally unifiable by θ and let σ be a most general unifier of $\mathbf{t}_{k+1}\theta$ and $\mathbf{u}_{k+1}\theta$. Then $\sigma \circ \theta$ is a most general unifier of $\langle \mathbf{t}_1, \ldots, \mathbf{t}_{k+1} \rangle$ and $\langle \mathbf{u}_1, \ldots, \mathbf{u}_{k+1} \rangle$.*

Proof: There are two things to prove: that $\sigma \circ \theta$ is a unifier and that it is most general. Notice that $\mathbf{t}_i\theta = \mathbf{u}_i\theta$ for $1 \leq i \leq k$ so $(\mathbf{t}_i\theta)\sigma = (\mathbf{u}_i\theta)\sigma$. Now $(\mathbf{t}_{k+1}\theta)\sigma = (\mathbf{u}_{k+1}\theta)\sigma$ because σ unifies $\mathbf{t}_{k+1}\theta$ and $\mathbf{u}_{k+1}\theta$.

We now show that $\sigma \circ \theta$ is most general. Let λ be any unifier of $\langle \mathbf{t}_1, \ldots, \mathbf{t}_{k+1} \rangle$ and $\langle \mathbf{u}_1, \ldots, \mathbf{u}_{k+1} \rangle$. The substitution λ is also a unifier of $\langle \mathbf{t}_1, \ldots, \mathbf{t}_k \rangle$ and $\langle \mathbf{u}_1, \ldots, \mathbf{u}_k \rangle$ so there is a substitution α such that $\lambda = \alpha \circ \theta$ (because θ is a most general unifier of $\langle \mathbf{t}_1, \ldots, \mathbf{t}_k \rangle$ and $\langle \mathbf{u}_1, \ldots, \mathbf{u}_k \rangle$). Now λ also unifies $\mathbf{t}_{k+1}$ and $\mathbf{u}_{k+1}$ so $\mathbf{t}_{k+1}\theta\alpha = \mathbf{u}_{k+1}\theta\alpha$ and hence α is a unifier of $\mathbf{t}_{k+1}\theta$ and $\mathbf{u}_{k+1}\theta$, so there is a substitution β such that $\alpha = \beta \circ \sigma$ (because σ is a most general unifier of $\mathbf{t}_{k+1}\theta$ and $\mathbf{u}_{k+1}\theta$). So $\lambda = (\beta \circ \sigma) \circ \theta = \beta \circ (\sigma \circ \theta)$ and $\sigma \circ \theta$ is indeed most general. ∎

If we wish to find a most general unifier for a set $\{\mathbf{A}_1, \ldots, \mathbf{A}_k\}$ of expressions we can look for a most general unifier of the two sequences $\langle \mathbf{A}_1, \ldots, \mathbf{A}_{k-1} \rangle$ and $\langle \mathbf{A}_2, \ldots, \mathbf{A}_k \rangle$. It is easy to see that a most general unifier of of the pair of sequences is a most general unifier of the set and vice versa. So to calculate the most general unifier of an arbitrary set of expressions it is sufficient to calculate the most general unifier of a pair of sequences of elements drawn from that set in the above way.

We now present the algorithm for calculating most general unifiers of two expressions.

Algorithm 4.2.1

$$\mathrm{mgu}(\mathbf{t},\mathbf{u}) = \begin{cases} \{\} & \textit{if } \mathbf{t}=\mathbf{u} \\ \{\mathbf{t}\mapsto\mathbf{u}\} & \textit{if } \mathbf{t} \textit{ is a variable and } \mathbf{t} \textit{ does not occur in } \mathbf{u}. \\ \{\mathbf{u}\mapsto\mathbf{t}\} & \textit{if } \mathbf{u} \textit{ is a variable, } \mathbf{t} \textit{ is not, and } \mathbf{u} \textit{ does not occur in } \mathbf{t} \\ \theta_n\circ\theta_{n-1}\circ\ldots\circ\theta_1 & \textit{if } \mathbf{t}=\mathbf{n}\mathbf{t}_1\ldots\mathbf{t}_n,\ \mathbf{u}=\mathbf{n}\mathbf{u}_1\ldots\mathbf{u}_n \textit{ and } \theta_i=\mathrm{mgu}(\mathbf{t}_i\theta_i\ldots\theta_{i-1},\mathbf{u}_i\theta_1\ldots\theta_{i-1}) \textit{ for } 1\le i\le n \\ \mathsf{FAIL} & \textit{otherwise.} \end{cases}$$

We need to show both that this algorithm halts and that it halts with a most general unifier of its inputs, when the inputs are unifiable, but first we require some preliminary results about the algorithm and the substitutions that it produces.

To clarify the number of steps that $\mathrm{mgu}(\mathbf{t},\mathbf{u})$ takes we define an additional function $\#\mathrm{mgu}(\mathbf{t},\mathbf{u})$ as follows:

$$\#\mathrm{mgu}(\mathbf{t},\mathbf{u}) = \begin{cases} 1 & \text{if } \mathbf{t}=\mathbf{u} \\ 1 & \text{if } \mathbf{t} \text{ is a variable and } \mathbf{t} \text{ does not occur in } \mathbf{u}. \\ 1 & \text{if } \mathbf{u} \text{ is a variable, } \mathbf{t} \text{ is not, and } \mathbf{u} \text{ does not occur in } \mathbf{t} \\ k_1+k_2+\cdots+k_n & \text{if } \mathbf{t}=\mathbf{n}\mathbf{t}_1\ldots\mathbf{t}_n,\ \mathbf{u}=\mathbf{n}\mathbf{u}_1\ldots\mathbf{u}_n \text{ and } k_i=\#\mathrm{mgu}(\mathbf{t}_i\theta_i\ldots\theta_{i-1},\mathbf{u}_i\theta_1\ldots\theta_{i-1}) \text{ for } 1\le i\le n \\ \mathsf{FAIL} & \text{otherwise.} \end{cases}$$

Where it is understood that the θ_is mentioned in the fourth case of the definition are the substitutions calculated in the corresponding case of the definition of mgu. When we refer to the number of steps in calculating $\mathrm{mgu}(\mathbf{t}, \mathbf{u})$ we mean $\#\mathrm{mgu}(\mathbf{t}, \mathbf{u})$.

Let $V(\mathbf{t})$ be the set of variables occurring in $\mathbf{t}$, for convenience let $V(\mathbf{t}, \mathbf{u}) = V(\mathbf{t}) \cup V(\mathbf{u})$, and let $v(\mathbf{t}, \mathbf{u})$ be the cardinality of $V(\mathbf{t}, \mathbf{u})$.[1] Further let $N(\mathbf{t})$ be the number of occurrences of non-logical symbols in $\mathbf{t}$ and let $n(\mathbf{t}, \mathbf{u}) = N(\mathbf{t}) + N(\mathbf{u})$. Then we define an ordering on pairs of expressions by: $\langle \mathbf{t}, \mathbf{u} \rangle < \langle \mathbf{t}', \mathbf{u}' \rangle$ if and only if either $v(\mathbf{t}, \mathbf{u}) < v(\mathbf{t}', \mathbf{u}')$ or both $v(\mathbf{t}, \mathbf{u}) = v(\mathbf{t}', \mathbf{u}')$ and $n(\mathbf{t}, \mathbf{u}) < n(\mathbf{t}', \mathbf{u}')$.

Lemma 4.2.2 *The ordering $<$ on pairs of expressions is irreflexive, transitive, and well-founded.*

Proof: Irreflexivity. Since $v(\mathbf{t}, \mathbf{u}) = v(\mathbf{t}, \mathbf{u})$ and $n(\mathbf{t}, \mathbf{u}) = n(\mathbf{t}, \mathbf{u})$ it is not possible that $\langle \mathbf{t}, \mathbf{u} \rangle < \langle \mathbf{t}, \mathbf{u} \rangle$.

Transitivity. Let $\langle \mathbf{t}, \mathbf{u} \rangle < \langle \mathbf{t}', \mathbf{u}' \rangle$ and let $\langle \mathbf{t}', \mathbf{u}' \rangle < \langle \mathbf{t}'', \mathbf{u}'' \rangle$. Since $\langle \mathbf{t}, \mathbf{u} \rangle < \langle \mathbf{t}', \mathbf{u}' \rangle$ either $v(\mathbf{t}, \mathbf{u}) < v(\mathbf{t}', \mathbf{u}')$ or both $v(\mathbf{t}, \mathbf{u}) = v(\mathbf{t}', \mathbf{u}')$ and $n(\mathbf{t}, \mathbf{u}) < n(\mathbf{t}', \mathbf{u}')$. If $v(\mathbf{t}, \mathbf{u}) < v(\mathbf{t}', \mathbf{u}')$ then $v(\mathbf{t}, \mathbf{u}) < v(\mathbf{t}'', \mathbf{u}'')$, and $\langle \mathbf{t}, \mathbf{u} \rangle < \langle \mathbf{t}'', \mathbf{u}'' \rangle$, because $v(\mathbf{t}', \mathbf{u}') \leq v(\mathbf{t}'', \mathbf{u}'')$. If, on the other hand, both $v(\mathbf{t}, \mathbf{u}) = v(\mathbf{t}', \mathbf{u}')$ and $n(\mathbf{t}, \mathbf{u}) < n(\mathbf{t}', \mathbf{u}')$ then either $v(\mathbf{t}', \mathbf{u}') < v(\mathbf{t}'', \mathbf{u}'')$, in which case $v(\mathbf{t}, \mathbf{u}) < v(\mathbf{t}'', \mathbf{u}'')$, or both $v(\mathbf{t}', \mathbf{u}') = v(\mathbf{t}'', \mathbf{u}'')$ and $n(\mathbf{t}', \mathbf{u}') < n(\mathbf{t}'', \mathbf{u}'')$, in which case $v(\mathbf{t}, \mathbf{u}) = v(\mathbf{t}'', \mathbf{u}'')$ and $n(\mathbf{t}, \mathbf{u}) < n(\mathbf{t}'', \mathbf{u}'')$. Either way $\langle \mathbf{t}, \mathbf{u} \rangle < \langle \mathbf{t}'', \mathbf{u}'' \rangle$ and the ordering is transitive.

Well-foundedness. Suppose $<$ is not well-founded; then there is an infinite descending chain of pairs $\langle \mathbf{t}_1, \mathbf{u}_1 \rangle > \langle \mathbf{t}_2, \mathbf{u}_2 \rangle > \cdots$. Since $v(\mathbf{t}_1, \mathbf{u}_1) \geq v(\mathbf{t}_2, \mathbf{u}_2) \geq \cdots$ there must be an m such that $v(\mathbf{t}_i, \mathbf{u}_i) = v(\mathbf{t}_m, \mathbf{u}_m)$ for all $i \geq m$. Since the sequence of pairs is descending $n(\mathbf{t}_{i+1}, \mathbf{u}_{i+1}) < n(\mathbf{t}_i, \mathbf{u}_i)$ for all $i \geq m$; this is clearly impossible. So $<$ is well-founded. ■

[1]Although v has been used as the name of a truth valuation previously we second it in this Chapter, everywhere else it still refers to a truth valuation.

A substitution θ is *proper* if whenever $\mathbf{x}$ occurs in $\mathbf{t}\theta$ then $\mathbf{x}\theta = \mathbf{x}$. The notation $\text{rng}^*(\theta)$ stands for the set $\{\mathbf{x}\theta | \mathbf{x}\theta \neq \mathbf{x}\}$ and by $V(\text{rng}^*(\theta))$ we mean the set of variables that occur in any of the expressions in $\text{rng}^*(\theta)$. In the arguments below the terms of crucial importantce are those of the form $\mathbf{x}\theta$ where $\mathbf{x}\theta$ differs from $\mathbf{x}$, this is what motivates the special notion of the range of a substitution $\text{rng}^*(\theta)$.

Lemma 4.2.3

1. $V(\text{rng}^*(\theta_2 \circ \theta_1)) \subseteq V(\text{rng}^*(\theta_1)) \cup V(\text{rng}^*(\theta_2))$.

2. $V(\mathbf{t}\theta, \mathbf{u}\theta) \subseteq V(\mathbf{t}, \mathbf{u}) \cup V(\text{rng}^*(\theta))$.

Proof: First part 1: let $\mathbf{x} \in V(\text{rng}^*(\theta_2 \circ \theta_1))$; thus $\mathbf{x}$ occurs in $\mathbf{y}\theta_1\theta_2$ and $\mathbf{y} \neq \mathbf{y}\theta_1\theta_2$, for some $\mathbf{y}$. Since $\mathbf{x}$ occurs in $(\mathbf{y}\theta_1)\theta_2$ there must be a variable, say $\mathbf{z}$, occurring in $\mathbf{y}\theta_1$, such that $\mathbf{x}$ occurs in $\mathbf{z}\theta_2$. There are two possibilities: $\mathbf{z}\theta_2 = \mathbf{z}$ or not. If $\mathbf{z}\theta_2 = \mathbf{z}$ then $\mathbf{z} = \mathbf{x}$ and since $\mathbf{z}$ occurred in $\mathbf{y}\theta_1$ the variable $\mathbf{x}$ occurs in $\mathbf{y}\theta_1$. Notice in this case that $\mathbf{x} \neq \mathbf{y}$ because otherwise $\mathbf{x} = \mathbf{y}\theta_1$ and $\mathbf{x} = \mathbf{x}\theta_1$ and $\mathbf{x}\theta_2 = \mathbf{x}$, so $\mathbf{x}\theta_1\theta_2 = \mathbf{x}$ and then $\mathbf{x} \notin V(\text{rng}^*(\theta_2 \circ \theta_1))$. So $\mathbf{x}$ is a member of $V(\text{rng}^*(\theta_1))$. If, on the other hand, $\mathbf{z}\theta_2 \neq \mathbf{z}$ then $\mathbf{x} \in V(\text{rng}^*(\theta_2))$.

Now part 2: let $\mathbf{x}$ occur in either $\mathbf{t}\theta$ or $\mathbf{u}\theta$; then there is a $\mathbf{z}$ occurring in either $\mathbf{t}$ or $\mathbf{u}$ such that $\mathbf{x}$ occurs in $\mathbf{z}\theta$. Either $\mathbf{z}\theta = \mathbf{z}$, in which case $\mathbf{z} = \mathbf{x}$ and $\mathbf{x} \in V(\mathbf{t}, \mathbf{u})$, or $\mathbf{z}\theta \neq \mathbf{z}$, in which case $\mathbf{x} \in V(\text{rng}^*(\theta))$. ∎

Lemma 4.2.4 *If* $\theta = \text{mgu}(\mathbf{t}, \mathbf{u})$ *then* $V(\text{rng}^*(\theta)) \subseteq V(\mathbf{t}, \mathbf{u})$.

Proof: The result is proved by induction on the length of a computation, recall that the length of the computation of $\text{mgu}(\mathbf{t}, \mathbf{u})$ is $\#\text{mgu}(\mathbf{t}, \mathbf{u})$.

Suppose the result holds for all computations of length less than k and that $\theta = \text{mgu}(\mathbf{t}, \mathbf{u})$ takes k steps. If $k = 1$ then θ is either $\{\}$, $\{\mathbf{t} \mapsto \mathbf{u}\}$ or $\{\mathbf{u} \mapsto \mathbf{t}\}$. In any case it is clear that the variables that occur in $\text{rng}^*(\theta)$ occur in either $\mathbf{t}$ or $\mathbf{u}$. Now suppose that $k > 1$; then $\mathbf{t}$ and $\mathbf{u}$ must be $\mathbf{pt}_1 \ldots \mathbf{t}_n$ and $\mathbf{pu}_1 \ldots \mathbf{u}_n$ respectively and $\theta = \theta_n \circ \cdots \circ \theta_1$, where $\theta_i = \text{mgu}(\mathbf{t}_i\theta_1 \ldots \theta_{i-1}, \mathbf{u}_i\theta_1 \ldots \theta_{i-1})$ for $1 \leq i \leq n$. Note that

$$V(\text{rng}^*(\theta_i)) \subseteq V(\mathbf{t}_i\theta_1 \ldots \theta_{i-1}, \mathbf{u}_i\theta_1 \ldots \theta_{i-1}) \quad \text{for } 1 \leq i \leq n$$

because the computation of $\text{mgu}(\mathbf{t}_i\theta_1 \ldots \theta_{i-1}, \mathbf{u}_i\theta_1 \ldots \theta_{i-1})$ requires fewer than k steps, for each $1 \leq i \leq n$. To see that $V(\text{rng}^*(\theta)) \subseteq V(\mathbf{t}, \mathbf{u})$ use Lemma 4.2.3(1) $n - 1$ times to obtain

$$V(\text{rng}^*(\theta_n \circ \cdots \circ \theta_1)) \subseteq V(\text{rng}^*(\theta_1)) \cup \ldots \cup V(\text{rng}^*(\theta_n))$$

and note that $V(\text{rng}^*(\theta_i)) \subseteq V(\mathbf{pt}_1 \ldots \mathbf{t}_i, \mathbf{pu}_1 \ldots \mathbf{u}_i)$ because

$$\theta_i = \text{mgu}(\mathbf{pt}_1 \ldots \mathbf{t}_i, \mathbf{pu}_1 \ldots \mathbf{u}_i), \quad \text{for } 1 \leq i \leq n.$$

It is clear that $V(\mathbf{pt}_1 \ldots \mathbf{t}_i, \mathbf{pu}_1 \ldots \mathbf{u}_i) \subseteq V(\mathbf{pt}_1 \ldots \mathbf{t}_n, \mathbf{pu}_1 \ldots \mathbf{u}_n)$, for $1 \leq i \leq n$, so $V(\text{rng}^*(\theta)) \subseteq V(\mathbf{t}, \mathbf{u})$. ■

Lemma 4.2.5 *If* $\theta = \text{mgu}(\mathbf{t}, \mathbf{u})$ *then* θ *is proper.*

Proof: This proof, like the previous, proceeds by an induction on the length of a computation.

Suppose that $\theta = \text{mgu}(\mathbf{t}, \mathbf{u})$ requires k steps to compute and that $\theta' = \text{mgu}(\mathbf{t}', \mathbf{u}')$ is proper for any computation requiring fewer than k steps. If $k = 1$ then the result holds by the requirements of Algorithm 4.2.3 in its first three cases. Consider a computation which has more than one step; then $\mathbf{t} = \mathbf{pt}_1 \ldots \mathbf{t}_n$, $\mathbf{u} = \mathbf{pu}_1 \ldots \mathbf{u}_n$, $\theta_i = \text{mgu}(\mathbf{t}_i\theta_1 \ldots \theta_{i-1}, \mathbf{u}_i\theta_1 \ldots \theta_{i-1})$ for $1 \leq i \leq n$, and $\theta = \theta_n \circ \cdots \circ \theta_1$. Since θ_n and

$\theta_{n-1} \circ \cdots \circ \theta_1$ both must require fewer than k steps to compute they must be proper. To demonstrate that $\theta_n \circ \cdots \circ \theta_1$ is proper let $\mathbf{x}$ occur in $\mathbf{w}\theta_1 \ldots \theta_n$ for some term $\mathbf{w}$.

Sub claim: $\mathbf{x}$ occurs in $\mathbf{v}\theta_1 \ldots \theta_{n-1}$ for some term $\mathbf{v}$. Towards a contradiction suppose that $\mathbf{x}$ does not occur in $\mathbf{v}\theta_1 \ldots \theta_{n-1}$ for any $\mathbf{v}$. Then, in particular $\mathbf{x}$ does not occur in either $\mathbf{t}_n\theta_1 \ldots \theta_{n-1}$ or $\mathbf{u}_n\theta_1 \ldots \theta_{n-1}$, i.e. $\mathbf{x} \notin V(\mathbf{t}_n\theta_1 \ldots \theta_{n-1}, \mathbf{u}_n\theta_1 \ldots \theta_{n-1})$. Because $\theta_n = \mathrm{mgu}(\mathbf{t}_n\theta_1 \ldots \theta_{n-1}, \mathbf{u}_n\theta_1 \ldots \theta_{n-1})$ and $\mathbf{x} \notin V(\mathbf{t}_n\theta_1 \ldots \theta_{n-1}, \mathbf{u}_n\theta_1 \ldots \theta_{n-1})$ it follows that $\mathbf{x} \notin V(\mathrm{rng}^*(\theta_n))$ by Lemma 4.2.4. So if $\mathbf{x}$ doesn't occur in $\mathbf{v}\theta_1 \ldots \theta_{n-1}$ and $\mathbf{x} \notin V(\mathrm{rng}^*(\theta_n))$ then $\mathbf{x}$ can not occur in $\mathbf{v}\theta_1 \ldots \theta_n$ for any $\mathbf{v}$ but this is clearly a contradiction because $\mathbf{x}$ occurs in $\mathbf{w}\theta_1 \ldots \theta_n$. So we have proved the sub claim and that $\mathbf{x}$ occurs in $\mathbf{v}\theta_1 \ldots \theta_{n-1}$ for some term $\mathbf{v}$.

Given that $\mathbf{x}$ occurs in $\mathbf{v}\theta_1 \ldots \theta_{n-1}$ for some $\mathbf{v}$ and that $\theta_{n-1} \circ \cdots \circ \theta_1$ is proper $\mathbf{x}\theta_1 \ldots \theta_{n-1} = \mathbf{x}$. Given that $\mathbf{x}$ occurs in $(\mathbf{w}\theta_1 \ldots \theta_{n-1})\theta_n$ and that θ_n is proper $\mathbf{x}\theta_n = \mathbf{x}$. So $\mathbf{x}\theta_1 \ldots \theta_n = \mathbf{x}$. ■

Lemma 4.2.6 *If* $\mathbf{t} = \mathbf{p}\mathbf{t}_1 \ldots \mathbf{t}_n$, $\mathbf{u} = \mathbf{p}\mathbf{u}_1 \ldots \mathbf{u}_n$, $1 \le j \le n$, *and*

$$\theta_i = \mathrm{mgu}(\mathbf{t}_i\theta_1 \ldots \theta_{i-1}, \mathbf{u}_i\theta_1 \ldots \theta_{i-1}), \quad \textit{for } 1 \le i \le j$$

then $\langle \mathbf{t}_i\theta_1 \ldots \theta_{i-1}, \mathbf{u}_i\theta_1 \ldots \theta_{i-1} \rangle < \langle \mathbf{t}, \mathbf{u} \rangle$ *for* $1 \le i \le j$.

Proof: Since $\theta_{i-1} \circ \cdots \circ \theta_1 = \mathrm{mgu}(\mathbf{p}\mathbf{t}_1 \ldots \mathbf{t}_{i-1}, \mathbf{p}\mathbf{u}_1 \ldots \mathbf{u}_{i-1})$ it follows that

$$V(\mathrm{rng}^*(\theta_{i-1} \circ \cdots \circ \theta_1)) \subseteq V(\mathbf{p}\mathbf{t}_1 \ldots \mathbf{t}_{i-1}, \mathbf{p}\mathbf{u}_1 \ldots \mathbf{u}_{i-1}),$$

by Lemma 4.2.4, and in turn

$$V(\mathbf{p}\mathbf{t}_1 \ldots \mathbf{t}_{i-1}, \mathbf{p}\mathbf{u}_1 \ldots \mathbf{u}_{i-1}) \subseteq V(\mathbf{t}, \mathbf{u}),$$

so $V(\mathrm{rng}^*(\theta_{i-1} \circ \cdots \circ \theta_1)) \subseteq V(\mathbf{t}, \mathbf{u})$. Clearly $V(\mathbf{t}_i, \mathbf{u}_i) \subseteq V(\mathbf{t}, \mathbf{u})$, so

$$V(\mathbf{t}_i\theta_1 \ldots \theta_{i-1}, \mathbf{u}_i\theta_1 \ldots \theta_{i-1}) \subseteq V(\mathbf{t}, \mathbf{u}),$$

by Lemma 4.2.3(2).

If $V(\mathbf{t}_i\theta_1 \ldots \theta_{i-1}, \mathbf{u}_i\theta_1 \ldots \theta_{i-1}) \subset V(\mathbf{t}, \mathbf{u})$ then $\langle \mathbf{t}_i\theta_1 \ldots \theta_{i-1}, \mathbf{u}_i\theta_1 \ldots \theta_{i-1}\rangle < \langle \mathbf{t}, \mathbf{u}\rangle$. On the other hand if $V(\mathbf{t}_i\theta_1 \ldots \theta_{i-1}, \mathbf{u}_i\theta_1 \ldots \theta_{i-1}) = V(\mathbf{t}, \mathbf{u})$ then, since $\theta_{i-1} \circ \cdots \circ \theta_1$ is proper (Lemma 4.2.5 with $\theta_{i-1} \circ \cdots \circ \theta_1 = \mathrm{mgu}(\mathbf{p}\mathbf{t}_1 \ldots \mathbf{t}_{i-1}, \mathbf{p}\mathbf{t}_1 \ldots \mathbf{t}_{i-1})$), $\mathbf{t}_i\theta_1 \ldots \theta_{i-1} = \mathbf{t}_i$ and $\mathbf{u}_i\theta_1 \ldots \theta_{i-1} = \mathbf{u}_i$ because if $\mathbf{x}$ occurs in either $\mathbf{t}_i$ or $\mathbf{u}_i$ and if $\mathbf{x}\theta_1 \ldots \theta_{i-1} \neq \mathbf{x}$ then $\mathbf{x}$ doesn't occur in either $\mathbf{t}_i\theta_1 \ldots \theta_{i-1}$ or $\mathbf{u}_i\theta_1 \ldots \theta_{i-1}$ but then

$$V(\mathbf{t}_i\theta_1 \ldots \theta_{i-1}, u_i\theta_1 \ldots \theta_{i-1}) \neq V(\mathbf{t}, \mathbf{u}).$$

Since $\mathbf{t}_i$ and $\mathbf{u}_i$ do not contain an occurrence of $\mathbf{p}$ that occurred in $\mathbf{t}$ and $\mathbf{u}$ it follows that $n(\mathbf{t}_i, \mathbf{u}_i) < n(\mathbf{t}, \mathbf{u})$ and $\langle \mathbf{t}_i\theta_1 \ldots \theta_{i-1}, \mathbf{u}_i\theta_1 \ldots \theta_{i-1}\rangle < \langle \mathbf{t}, \mathbf{u}\rangle$. ∎

Theorem 4.2.1 (The Unification Theorem) *The algorithm* $\mathrm{mgu}(\mathbf{t}, \mathbf{u})$ *halts; and* $\mathbf{t}$ *and* $\mathbf{u}$ *are unifiable if and only if* $\theta = \mathrm{mgu}(\mathbf{t}, \mathbf{u})$ *is a most general unifier of* $\mathbf{t}$ *and* $\mathbf{u}$.

Proof: Suppose that for some pair of expressions the algorithm does not halt. Then there must be a least pair, with respect to the ordering $<$ of pairs of expressions, $\langle \mathbf{t}, \mathbf{u}\rangle$ say, for which $\mathrm{mgu}(\mathbf{t}, \mathbf{u})$ does not halt. Because $\mathrm{mgu}(\mathbf{t}, \mathbf{u})$ does not halt, $\mathbf{t}$ must be $\mathbf{l}\mathbf{t}_1 \ldots \mathbf{t}_n$ and $\mathbf{u}$ must be $\mathbf{l}\mathbf{u}_1 \ldots \mathbf{u}_n$ and there must be a least k, $1 \leq k \leq n$, for which $\mathrm{mgu}(\mathbf{t}_k\theta_1 \ldots \theta_{k-1}, \mathbf{u}_k\theta_1 \ldots \theta_{k-1})$ does not halt and $\theta_1, \ldots, \theta_{k-1}$ are $\mathrm{mgu}(\mathbf{t}_1, \mathbf{u}_1), \ldots, \mathrm{mgu}(\mathbf{t}_{k-1}\theta_1 \ldots \theta_{k-2}, \mathbf{u}_{k-1}\theta_1 \ldots \theta_{k-2})$. By Lemma 4.2.6

$$\langle \mathbf{t}_k\theta_1 \ldots \theta_{k-1}, \mathbf{u}_k\theta_1 \ldots \theta_{k-1}\rangle < \langle \mathbf{t}, \mathbf{u}\rangle,$$

but this contradicts the leastness of $\langle \mathbf{t}, \mathbf{u}\rangle$.

Now that we have shown that $\mathrm{mgu}(\mathbf{t}, \mathbf{u})$ must halt, we still need to verify that when $\theta = \mathrm{mgu}(\mathbf{t}, \mathbf{u})$ then θ is a unifier and in fact is a most general unifier.

Case 1: $\mathbf{t} = \mathbf{u}$. In this case $\theta = \{\} = \text{mgu}(\mathbf{t}, \mathbf{u})$. Any unifier λ of $\mathbf{t}$ and $\mathbf{u}$ can be expressed as $\lambda \circ \{\}$ so $\{\}$ is a most general unifier.

Case 2: $\mathbf{t}$ is a variable and $\mathbf{u} \neq \mathbf{t}$. In this case $\theta = \{\mathbf{t} \mapsto \mathbf{u}\}$. Note that the variable $\mathbf{t}$ does not occur in $\mathbf{u}$. Let λ be any unifier of $\mathbf{t}$ and $\mathbf{u}$, note that $\mathbf{t}\theta = \mathbf{u}$ so $\mathbf{t}\theta\lambda = \mathbf{u}\lambda = \mathbf{u}\theta\lambda$ and so $\lambda = \lambda \circ \theta$ and hence θ is most general.

Case 3: $\mathbf{u}$ is a variable and $\mathbf{t}$ is not. This case is analogous to Case 2.

Case 4: $\mathbf{t} = \mathbf{l}\mathbf{t}_1 \ldots \mathbf{t}_n$, $\mathbf{u} = \mathbf{l}\mathbf{u}_1 \ldots \mathbf{u}_n$, and $\theta_i = \text{mgu}(\mathbf{t}_i\theta_1 \ldots \theta_{i-1}, \mathbf{u}_i\theta_1 \ldots \theta_{i-1})$ for $1 \leq i \leq n$. We need to show that $\theta_n \circ \ldots \circ \theta_1$ is a most general unifier of $\langle \mathbf{t}_1, \ldots, \mathbf{t}_n \rangle$ and $\langle \mathbf{u}_1, \ldots, \mathbf{u}_n \rangle$ but this is clear by induction using Lemma 4.2.1.

Finally we show that if $\text{mgu}(\mathbf{t}, \mathbf{u})$ returns a FAIL result then $\mathbf{t}$ and $\mathbf{u}$ are not unifiable. Suppose to the contrary that their is a pair of expressions that are unifiable but the algorithm fails to produce a unifier. Now since there is at least one such pair let the pair $\langle \mathbf{t}, \mathbf{u} \rangle$ be least such that $\mathbf{t}$ and $\mathbf{u}$ are unifiable but $\text{mgu}(\mathbf{t}, \mathbf{u}) = \mathsf{FAIL}$. If $\mathbf{t}$ or $\mathbf{u}$ is a variable then there is just one way for the algorithm to fail, that is for $\mathbf{t}$ to occur in $\mathbf{u}$ (or the other way around, the cases are analogous) and $\mathbf{t} \neq \mathbf{u}$. If this is the case any would-be unifier, θ say, would map $\mathbf{t}$ to some term $\mathbf{v}$ but then $\mathbf{v}$ would occur within $\mathbf{u}\theta$ so θ couldn't possibly unify $\mathbf{t}$ and $\mathbf{u}$. The other case in which the algorithm can produce a FAIL result is if in the calculation of $\text{mgu}(\mathbf{t}, \mathbf{u})$ we must first calculate $\text{mgu}(\mathbf{t}_i\theta_1 \ldots \theta_{i-1}, \mathbf{t}_i\theta_1 \ldots \theta_{i-1})$ and the later calculation results in a FAIL result but this is clearly impossible by our choice of $\langle \mathbf{t}, \mathbf{u} \rangle$ because $\langle \mathbf{t}_i\theta_1 \ldots \theta_{i-1}, \mathbf{t}_i\theta_1 \ldots \theta_{i-1} \rangle < \langle \mathbf{t}, \mathbf{u} \rangle$, and $\langle \mathbf{t}, \mathbf{u} \rangle$ was supposed to be least. ∎

Chapter 5

Resolution

In this chapter we present the resolution procedure and show that for any particular set S of clauses this procedure will demonstrate the unsatisfiability of S if S is unsatisfiable. We also live up to our promise to demonstrate how unification calculates a proof set.

5.1 The Resolution Procedure

The resolution procedure is similar to the ground resolution procedure. In fact, all we must do is extend the definition of resolvant so that we may calculate the resolvant of two arbitrary clauses, not just ground clauses.

Definition 5.1.1 *Let C and D be two clauses; also let L and M be such that $L \subseteq C$ and $M \subseteq D$. Further let N be the set of atomic formulas $\mathbf{A}$ such that either $\mathbf{A}$ or the complement $\mathbf{A}'$ is a member of $L\xi_C \cup M\eta_D$. Also suppose that N is most* *generally unifiable by σ_N and that $L\xi_C\sigma_N$ and $M\eta_D\sigma_N$ are singleton sets containing complementary literals; then*

$$(C - L)\xi_C\sigma_N \cup (D - M)\eta_D\sigma_N$$

is a resolvant *of the clauses C and D. If S is a set of clauses then $\mathcal{R}(S)$, the* resolution *of S, is S together with all possible resolvants of every pair of clauses in S. We also define the n-*resolution *$\mathcal{R}^n(S)$ inductively by: $\mathcal{R}^0(S) = S$ and $\mathcal{R}^n(S) = \mathcal{R}(\mathcal{R}^{n-1}(S))$.*

Note that in this definition the role of of the x- and y-standardizations, ξ_C and η_D, is simply to ensure that the variables that occur in $C\xi_C$ and $D\eta_D$ are disjoint. Ensuring that the variables are disjoint guarantees that we do not mistakenly bind logically different variables to the same understood universal quantifier.

Given these definitions we may go on to describe the resolution procedure. To demonstrate the validity of a sentence **A** we:

1. Convert $\neg$**A** to a set S of clauses, as outlined in Section 2.3.
2. Let $i \leftarrow 0$.
3. While $\Box \notin \mathcal{R}^i(S)$ do $i \leftarrow i + 1$.

If we manage to find an i for which $\Box \in \mathcal{R}^i(S)$ then we have demonstrated the unsatisfiability of the set S of clauses and hence the validity of **A**.

Notice that the resolution procedure does not constitute a decision procedure for first-order logic, i.e. if **A** is not valid, then the procedure will never halt.

Before we justify the resolution procedure the following example should make the reader reasonably familiar with the way the algorithm works.

Example 5.1.1 *Every group of exponent two is Abelian.*

The example is from group theory, actually a fragment of group theory—we will not require axioms about inverses. We show, with the standard axioms of group theory, that: every group of exponent two is abelian, i.e. if $x^2 = e$ for every element x of the group then $y \cdot z = z \cdot y$ for every y and z in the group. First we formalize group theory in a first order language. Our language consists of a 3-ary predicate symbol G and a constant symbol e. The idea is that $G(x, y, z)$ should represent $x \cdot y = z$, and that e is the group identity.

The associativity axioms of group theory are then represented by:

$$G(x,y,u) \wedge G(y,z,v) \wedge G(u,z,w) \to G(x,v,w)$$
$$G(x,y,u) \wedge G(y,z,v) \wedge G(x,v,w) \to G(u,z,w).$$

Now we represent the identity axioms as:

$$G(x,e,x)$$
$$G(e,x,x).$$

The axiom $G(x,x,e)$ represents the fact that the group is of exponent two. Finally we can represent $x \cdot y = y \cdot x$ by

$$G(x,y,z) \to G(y,x,z).$$

The sentence **A** we are attempting to prove by the resolution procedure is an implication whose antecedent is a conjunction of the associativity axioms, identity axioms, and axioms expressing that the group is of exponent two. The consequent of **A** is $G(x,y,z) \to G(y,x,z)$. After preparing the sentence $\neg\mathbf{A}$ as indicated in Section 2.3 we obtain a conjunction of the associativity and identity axioms together with $G(a,b,c) \wedge \neg G(b,a,c)$, the Skolemized negation of the consequent. So the set of clauses on which we perform resolution is

$$\langle\ [\neg G(x,y,u), \neg G(y,z,v), \neg G(u,z,w), G(x,v,w)] \tag{5.1}$$

$$[\neg G(x,y,u), \neg G(y,z,v), \neg G(x,v,w), G(u,z,w)] \tag{5.2}$$

$$[G(x,e,x)] \tag{5.3}$$

$$[G(e,x,x)] \tag{5.4}$$

$$[G(x,x,e)] \tag{5.5}$$

$$[G(a,b,c)] \tag{5.6}$$

$$[\neg G(b,a,c)] \qquad\qquad \rangle \tag{5.7}$$

In this example we will not calculate all possible resolvants of the clauses at each stage; this would generate far too many clauses to serve as a illuminating example. Instead we will demonstrate a sequence of resolution steps that results in the empty clause. We will also not explicitly form both the x- and y-standardizations of the clauses we are resolving upon. We instead will ensure that the variables that occur in each clause are disjoint from the variables occurring in the other clause. We present a resolution step in the following way

$$\begin{array}{l} C \\ D \\ \underline{\hspace{3cm}}\sigma_N \\ (C-L)\xi_C\sigma_N \cup (D-M)\eta_D\sigma_N \end{array}$$

underlining the atomic formulas in C and D which comprise the subsets L and M in the definition of resolution, Definition 5.1.1.

Resolving the x-standardization of (1) with (7)

$$[\neg G(x_1,x_2,x_4),\neg G(x_2,x_3,x_5),\neg G(x_4,x_3,x_6),\underline{G(x_1,x_5,x_6)}]$$
$$[\underline{\neg G(b,a,c)}]$$
$$\overline{}\{x_1\mapsto b,x_5\mapsto a,x_6\mapsto c\}$$
$$[\neg G(b,x_2,x_4),\neg G(x_2,x_3,a),\neg G(x_4,x_3,c)].$$

And resolving this with the y-standardization of (5)

$$[\underline{\neg G(b,x_2,x_4)},\neg G(x_2,x_3,a),\neg G(x_4,x_3,c)]$$
$$[\underline{G(y_1,y_1,e)}]$$
$$\overline{}\{y_1\mapsto b,x_2\mapsto b,x_4\mapsto e\}$$
$$[\neg G(b,x_3,a),\neg G(e,x_3,c)].$$

And resolving this with the y-standardization of (4)

$$[\neg G(b,x_3,a),\underline{\neg G(e,x_3,c)}]$$
$$[\underline{G(e,y_1,y_1)}]$$
$$\overline{}\{x_3\mapsto c,y_1\mapsto c\}$$
$$[\neg G(b,c,a)].$$

Now we resolve this with the x-standardization of (2)

$$[\neg G(x_1,x_2,x_3),\neg G(x_2,x_4,x_5),\neg G(x_1,x_5,x_6),\underline{G(x_3,x_4,x_6)}]$$
$$[\underline{\neg G(b,c,a)}]$$
$$\overline{}\{x_3\mapsto b,x_4\mapsto c,x_6\mapsto a\}$$
$$[\neg G(x_1,x_2,b),\neg G(x_2,c,x_5),\neg G(x_1,x_5,a)].$$

And then this with the y-standardization of (5)

$$[\neg G(x_1,x_2,b),\underline{\neg G(x_2,c,x_5)},\neg G(x_1,x_5,a)]$$
$$[\underline{G(y_1,y_1,e)}]$$
$$\overline{}\{x_2\mapsto c,y_1\mapsto c,x_5\mapsto e\}$$
$$[\neg G(x_1,c,b),\neg G(x_1,e,a)].$$

And this with the y-standardization of (3),

$$\begin{array}{l}
[\neg G(x_1,c,b),\underline{\neg G(x_1,e,a)}] \\
[\underline{G(y_1,e,y_1)}] \\
\rule{2em}{0.4pt}\{x_1 \mapsto a, y_1 \mapsto a\} \\
[\neg G(a,c,b)].
\end{array}$$

Now we resolve this with the x-standardization of (1),

$$\begin{array}{l}
[\neg G(x_1,x_2,x_4),\neg G(x_2,x_3,x_5),\neg G(x_4,x_3,x_6),\underline{G(x_1,x_5,x_6)}] \\
[\underline{\neg G(a,c,b)}] \\
\rule{10em}{0.4pt}\{x_1 \mapsto a, x_5 \mapsto c, x_6 \mapsto b\} \\
[\neg G(a,x_2,x_4),\neg G(x_2,x_3,c),\neg G(x_4,x_3,b)].
\end{array}$$

And this with (5)

$$\begin{array}{l}
[\underline{\neg G(a,x_2,x_4)},\neg G(x_2,x_3,c),\neg G(x_4,x_3,b)] \\
[\underline{G(y_1,y_1,e)}] \\
\rule{5em}{0.4pt}\{x_2 \mapsto a, x_4 \mapsto e, y_1 \mapsto a\} \\
[\neg G(a,x_3,c),\neg G(e,x_3,b)].
\end{array}$$

And this in turn with (4)

$$\begin{array}{l}
[\neg G(a,x_3,c),\underline{\neg G(e,x_3,b)}] \\
[\underline{G(e,y_1,y_1)}] \\
\rule{2em}{0.4pt}\{x_3 \mapsto b, y_1 \mapsto b\} \\
[\neg G(a,b,c)].
\end{array}$$

Finally this with (6)

$$\begin{array}{l}
[\underline{\neg G(a,b,c)}] \\
[\underline{G(a,b,c)}] \\
\rule{2em}{0.4pt}\{\} \\
\square.
\end{array}$$

5.2 The Resolution Theorem

Here we provide the proof that justifies the resolution procedure of the preceding section; the result is in [Robinson65].

Theorem 5.2.1 (The Resolution Theorem) *Let S be a set of clauses and P be a subset of* Herb(S). *Then $\mathcal{R}(P(S)) \subseteq P(\mathcal{R}(S))$.*

Proof: Let $A \in \mathcal{R}(P(S))$; then either $A \in P(S)$, in which case $A \in P(\mathcal{R}(S))$ because $S \subseteq \mathcal{R}(S)$, or A is a ground resolvant of two clauses, say C and D, in $P(S)$. Now $C = C'\alpha$ and $D = D'\beta$ where C' and D' are both elements of S and α and β are the substitutions $\{\mathbf{x}_1 \mapsto \mathbf{t}_1, \ldots \mathbf{x}_n \mapsto \mathbf{t}_n\}$ and $\{\mathbf{y}_1 \mapsto \mathbf{u}_1, \ldots \mathbf{y}_m \mapsto \mathbf{u}_m\}$, where $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are all the variables occurring in C' and $\mathbf{y}_1, \ldots, \mathbf{y}_m$ are all the variables occurring in D' and further that $\mathbf{t}_1, \ldots, \mathbf{t}_n, \mathbf{u}_1, \ldots, \mathbf{u}_m$ are all elements of P.

Since A is a ground resolvant of C and D there are literals $\mathbf{A} \in C$ and $\mathbf{B} \in D$ that are complements and $A = (C - \{\mathbf{A}\}) \cup (D - \{\mathbf{B}\})$. Note that $\mathbf{A} \in C'\alpha$ and $\mathbf{B} \in D'\beta$ so there are sets $L \subseteq C'$ and $M \subseteq D'$ such that $L\alpha = \{\mathbf{A}\}$ and $M\beta = \{\mathbf{B}\}$.

Let $\theta = \{x_1 \mapsto \mathbf{t}_1, \ldots, x_n \mapsto \mathbf{t}_n, y_1 \mapsto \mathbf{u}_1, \ldots, y_m \mapsto \mathbf{u}_m\}$; then we can see that $A = (C' - L)\xi_{C'}\theta \cup (D' - M)\eta_{D'}\theta$, where $\xi_{C'} = \{\mathbf{x}_1 \mapsto x_1, \ldots, \mathbf{x}_n \mapsto x_n\}$ and $\eta_{D'} = \{\mathbf{y}_1 \mapsto y_1, \ldots, \mathbf{y}_m \mapsto y_m\}$. It is clear that θ is a unifier of N the set of atomic formulas such that they or their complements occurs in $L\xi_{C'} \cup M\eta_{D'}$. So there is a most general unifier σ_N of N and $\theta = \lambda \circ \sigma_N$ for some substitution λ. Now let $B = (C' - L)\xi_{C'}\sigma_n \cup (D' - M)\eta_{D'}\sigma_n$ and note that B is a resolvant of C' and D' so $B \in \mathcal{R}(S)$. Notice that $A = B\lambda$ and that λ maps all of the variables that occur in B to elements of P so $A \in P(\mathcal{R}(S))$. ∎

Corollary 5.2.1 $\mathcal{R}^n(P(S)) \subseteq P(\mathcal{R}^n(S))$.

Proof: The Resolution Theorem is the $n = 1$ case. Now suppose that $\mathcal{R}^k(P(S)) \subseteq P(\mathcal{R}^k(S))$ and consider $\mathcal{R}^{k+1}(P(S))$. Now

$$\mathcal{R}^{k+1}(P(S)) = \mathcal{R}(\mathcal{R}^k(P(S)))$$

and $\mathcal{R}(\mathcal{R}^k(P(S)) \subseteq \mathcal{R}(P(\mathcal{R}^k(S)))$ because of the induction hypothesis and the easily verifiable fact: $A \subseteq B \Rightarrow \mathcal{R}(A) \subseteq \mathcal{R}(B)$. Now

$$\mathcal{R}(P(\mathcal{R}^k(S))) \subseteq P(\mathcal{R}(\mathcal{R}^k(S)))$$

because of the Resolution Theorem so $\mathcal{R}^{k+1}(P(S)) \subseteq P(\mathcal{R}^{k+1}(S))$. ∎

Theorem 5.2.2 (Completeness for Resolution Theorem Proving) *If a set of clauses S is unsatisfiable then $\Box \in \mathcal{R}^n(S)$ for some $n < \omega$.*

Proof: If S is unsatisfiable, then by Lemma 3.3.2 there exists a finite $P \subseteq \text{Herb}(S)$ and an $n < \omega$ such that $\Box \in \mathcal{R}^n(P(S))$. By application of Corollary 5.2.1 to $\Box \in \mathcal{R}^n(P(S))$ we obtain $\Box \in P(\mathcal{R}^n(S))$. Notice that if $\Box \in P(\mathcal{R}^n(S))$ then $\Box$ must be a member of $\mathcal{R}^n(S)$. ∎

Theorem 5.2.3 (Soundness of the Resolution Rule) *If $\models_{\mathfrak{A}} C$, $\models_{\mathfrak{A}} D$, and R is a resolvant of C and D then $\models_{\mathfrak{A}} R$.*

Proof: Since R is equivalent to a universal sentence, if we show that $\mathfrak{A}$ satisfies every ground instance of R then we will have shown that $\models_{\mathfrak{A}} R$. Notice that each ground instance of R can be obtained by applying a substitution, λ say, to R. Thus if $\models_{\mathfrak{A}} R\lambda$ for every substitution λ then $\models_{\mathfrak{A}} R$.

Let $\mathfrak{A}$ make both C and D true; also let λ be an arbitrary substitution such that $R\lambda$ is a ground clause. From the definition of a resolvant

$$R = (C - L)\xi_C\sigma_N \cup (D - M)\eta_D\sigma_N.$$

Since $\mathfrak{A}$ makes C and D true it also makes $C\xi_C\sigma_N$ and $D\eta_D\sigma_N$ true; further $\mathfrak{A}$ makes $C\xi_C\sigma_N\lambda$ and $D\eta_D\sigma_N\lambda$ true as well. It may happen that $C\xi_C\sigma_N\lambda$ and $D\eta_D\sigma_N\lambda$ are not ground clauses so select an arbitrary substitution, ν say, such that $C\xi_C\sigma_N\lambda\nu$ and $D\eta_D\sigma_N\lambda\nu$ are ground clauses. It is easy to see that $R\lambda$ is a ground resolvant of $C\xi_C\sigma_N\lambda\nu$ and $D\eta_D\sigma_N\lambda\nu$. Proposition 3.2.1 now produces that $\models_{\mathfrak{A}} R\lambda$, but recall that λ was arbitrary so this holds for all substitutions λ such that $R\lambda$ is a ground clause. So we may now conclude that $\models_{\mathfrak{A}} R$, because $\mathfrak{A}$ satisfies every ground instance of R. ■

5.3 The Role of Unification in Resolution

Recall that a proof set P for a set S of clauses is a set of ground terms, in the language of S, such that the Ground Resolution Procedure terminates after n steps with $\Box \in \mathcal{R}^n(P(S))$. It was previously mentioned that unification calculates a proof set for a particular set of clauses; this section is intended to justify that claim. But, we must first introduce the notion of a deduction.

Definition 5.3.1 *A* deduction (ground deduction) *of a clause C_n from a set S of clauses is a sequence $C_1, C_2, \ldots, C_n$ of clauses such that each C_i is either an element of S or a resolvant (ground resolvant) of C_j and C_k, for $1 \leq j, k < i$. A deduction (ground deduction) from S ending in $\Box$ is a* refutation (ground refutation) *of S.*

Theorem 5.3.1 *There is a deduction of C from S if and only if $C \in \mathcal{R}^n(S)$ for some $n < \omega$.*

Proof: ($\Rightarrow$) Let $C_0, \ldots, C_n = C$ be a deduction of C from S. Now we show that $C_i \in \mathcal{R}^i(S)$, by induction on i. Clearly $C_0 \in S$ and so $C_0 \in \mathcal{R}^0(S)$. Suppose that C_i

is in $\mathcal{R}^i(S)$. Since C_{i+1} is either an element of S, in which case $C_{i+1} \in \mathcal{R}^{i+1}(S)$, or C_{i+1} is a resolvant of C_j and C_k with $1 \leq j, k < i+1$, in which case $C_{i+1} \in \mathcal{R}^{i+1}(S)$. So $C_1, \ldots, C_{i+1} \in \mathcal{R}^{i+1}(S)$.

($\Leftarrow$) We use induction on the number of applications of $\mathcal{R}$. If $C \in \mathcal{R}^0(S)$ then $C \in S$ and hence the sequence C is a deduction of C from S. Now suppose that $D \in \mathcal{R}^m(S)$ implies that there is a deduction: $D_1, \ldots, D_m = D$. Let C be an element of $\mathcal{R}^{m+1}(S)$ that is not in $\mathcal{R}^m(S)$, thus C must be a resolvant of two elements D' and D'' of $\mathcal{R}^m(S)$. Since D' and D'' are elements of $\mathcal{R}^m(S)$ there are deductions: $D'_1, \ldots, D'_m = D'$ and $D''_1, \ldots, D''_l = D''$, but then $D'_1, \ldots, D'_m, D''_1, \ldots, D''_l, C$ is a deduction of C. ∎

Corollary 5.3.1 *There is a refutation of S if and only if $\Box \in \mathcal{R}^n(S)$ for some $n < \omega$.*

Let P be a set of terms; a P-instance of an expression $\mathbf{t}$ is $\mathbf{t}\lambda$ where λ is a substitution with $\text{rng}(\lambda) \subseteq P$. A P-instance of a clause C is $\{A\lambda | A \in C\}$ where $\text{rng}(\lambda) \subseteq P$. A P-instance of a set S of clauses is $\{C\lambda | C \in S\}$ with $\text{rng}(\lambda) \subseteq P$. Notice that the saturation of a set S of clauses with P is the set of all P-instances of every clause of S, specifically $P(S) = \{C\lambda | C \in S, \text{rng}(\lambda) \subseteq P\}$.

Let $C_1, \ldots, C_n = \Box$ be a refutation from S and let $\sigma_1, \ldots, \sigma_n$ be substitutions such that σ_i is the identity function if $C_i \in S$, and if C_i is a resolvant of C_j and C_k, where

$$C_i = (C_j - L)\xi_{C_j}\sigma_N \cup (D_k - M)\eta_{C_k}\sigma_N$$

and $1 \leq j, k < i$, then σ_i is σ_N.

We obtain a proof set for S is the following way. First, if the language of S has no constant symbols, augment the language with a new constant symbol. Second, let $P_n = \{\mathbf{a}\}$, where $\mathbf{a}$ is any constant symbol from the language of S. Finally, for $i < n$

let

$$P_i = \{\mathbf{t} | \mathbf{t} = \mathbf{x}\sigma_{i+1}\lambda \text{ for some variable } \mathbf{x} \text{ and some } \lambda \text{ such that } \mathrm{rng}(\lambda) \subseteq P_{i+1}\}.$$

We claim that P_1 is the required proof set for S. Observe that $P_j \subseteq P_i$ for $1 \leq i < j \leq n+1$.

To show that P_1 is a proof set we exhibit a ground refutation from $P_1(S)$. Let Σ_i be a sequence comprising all of the P_i-instances of the C_i. We need to show that the sequence obtained by concatenating sequences Σ_1 through Σ_n, we denote this concatenation of sequences as $\Sigma_1, \ldots, \Sigma_n$, is a ground refutation, i.e. $\Sigma_n = \Box$. We show this sequence is a ground refutation by induction on the length n of the resolution refutation of S. Observe that $C_1 \in S$; hence any element of Σ_1 is in $P_1(S)$ so Σ_1 is a ground deduction. Now suppose that $\Sigma_1, \ldots, \Sigma_k$ is a ground deduction and let C'_{k+1} be any element of Σ_{k+1}. Since C'_{k+1} is a P_{k+1}-instance of C_{k+1} then $C'_{k+1} = C_{k+1}\lambda$, with $\mathrm{rng}(\lambda) \subseteq P_{k+1}$. If $C_{k+1} \in S$ then $C_{k+1}\lambda \in P_{k+1}(S) \subseteq P_1(S)$ and hence $C'_{k+1} \in P_1(S)$, so $\Sigma_1, \ldots, \Sigma_{k+1}, C_{k+1}\lambda$ is ground deduction of $C_{k+1}\lambda$ from $P_1(S)$. If $C_{k+1} \notin S$ then

$$C_{k+1} = (C_l - L)\xi_{C_l}\sigma_{k+1} \cup (C_m - M)\eta_{C_m}\sigma_{k+1}$$

and

$$C_{k+1}\lambda = (C_l - L)\xi_{C_l}\sigma_{k+1}\lambda \cup (C_m - M)\eta_{C_m}\sigma_{k+1}\lambda.$$

To conclude the argument we show that C'_{k+1} is a ground resolvant of $C_l\xi_{C_l}\sigma_{k+1}\lambda$ and $C_m\eta_{C_m}\sigma_{k+1}\lambda$ and that $C_l\xi_{C_l}\sigma_{k+1}\lambda$ and $C_m\eta_{C_m}\sigma_{k+1}\lambda$ occur in Σ_l and Σ_m respectively.

Because of σ_{k+1}'s action on L and M ($L\xi_{C_l}\sigma_{k+1}$ and $M\eta_{C_m}\sigma_{k+1}$ are complementary singletons) and because λ makes any clause to which it is applied a ground clause $C_{k+1}\lambda$ is indeed a ground resolvant of $C_l\xi_{C_l}\sigma_{k+1}\lambda$ and $C_m\eta_{C_m}\sigma_{k+1}\lambda$.

It remains to show that $\mathrm{rng}(\lambda \circ \sigma_{k+1} \circ \xi_{C_l}) \subseteq P_l$, for if this is the case then $C_l \xi_{C_l} \sigma_{k+1} \lambda$ occurs in the sequence Σ_l. Let $\mathbf{t} \in \mathrm{rng}(\lambda \circ \sigma_{k+1} \circ \xi_{C_l})$; since ξ_{C_l} only renames variables this is the same as saying that $\mathbf{t} = \mathbf{x}\sigma_{k+1}\lambda$, for some variable $\mathbf{x}$. Observe that $\mathrm{rng}(\lambda) \subseteq P_{k+1}$ so $\mathbf{t} = \mathbf{x}\sigma_{k+1}\lambda$ is in P_k, by the definition of P_k. Since $l < k+1$ the term $\mathbf{t}$ is also a member of P_l. We have demonstrated that $C_l \xi_{C_l} \sigma_{k+1} \lambda$ occurs in Σ_l. Similarly $C_m \eta_{C_m} \sigma_{k+1} \lambda$ occurs in Σ_m.

Since the choice of C'_{k+1} was arbitrary we in fact have that $\Sigma_1, \ldots, \Sigma_{k+1}$ is a deduction from $P_1(S)$.

Finally, since $C_n = \Box$ and any P_n-instance of $\Box$ is $\Box$, the ground deduction $\Sigma_1, \ldots, \Sigma_n$ is a ground refutation of $P_1(S)$.

5.4 Conclusion

We conclude this thesis with an example that demonstrates that Resolution is indeed an improvement over the saturation methods described at the end of Chapter 3.

For this particular example we drop the parentheses for functions, with the understanding that the reader could readily provide them if required. We consider the following set of clauses:

$$\langle [P(a)], [\neg P(x), P(fx)], [\neg P(f^{2^n}a)] \rangle.$$

First we will demonstrate the invalidity of this set of clauses with Resolution.

Resolving the second clause with itself, for example yields

$$\begin{array}{l}[\neg P(x), P(fx)] \\ [\neg P(y), P(fy)] \\ \text{---}\{x \mapsto fy\} \\ [\neg P(y), P(ffy)]\end{array}$$

and then resolving this resolvant with itself yields

$$\begin{array}{l}[\neg P(x), P(ffx)] \\ [\neg P(y), P(ffy)] \\ \text{----}\{x \mapsto fy\} \\ [\neg P(y), P(ffffy)].\end{array}$$

It is easily seen that there is a deduction of length $n+1$:

$$[\neg P(x), P(fx)], [\neg P(x), P(ffx)], \ldots, [\neg P(x), P(f^{2^n}x)]$$

where each clause in the deduction is a resolvant of two copies of the immediately prior clause, one of which has had all of its variables renamed y. Also note that in each case the substitution is of the form $\{y \mapsto f^{2^k}(x)\}$. Hence there is a refutation of length $n+5$:

$$\begin{array}{c}[\neg P(x), P(fx)], [\neg P(x), P(ffx)], \ldots, [\neg P(x), P(f^{2^n}x)], \\ [P(a)], [P(f^{2^n}a)], [\neg P(f^{2^n}a)], \square.\end{array}$$

Consider a ground resolution proof from the same set of clauses. The first thing to notice is that because we must saturate the set of clauses we are interested in before we start producing ground resolvants we will be unable to achieve the same "doubling up" that occurred in the resolution case. We require clauses of the form $[\neg P(f^k a), P(f^{k+1}a)]$ for $0 \leq k \leq 2^k - 1$. A refutation then takes on the form:

$$\begin{array}{l}[\neg P(a), P(fa)], \quad [\neg P(fa), P(ffa)], [\neg P(a), P(ffa)], [\neg P(ffa) P(fffa)], \\ \qquad \ldots, [\neg P(a) P(f^{2^n}a)], [P(a)], [P(f^{2^n}a)], [\neg P(f^{2^n}a)], \square.\end{array}$$

This refutation is of length $2^{n+1}+3$. It is clear that resolution is an improvement over ground resolution in terms of length of refutation, at least in this particular case. And, by Corollary 5.2.1 it is also clear that resolution could not have a longer shortest refutation.

With the method of the last section we can calculate a proof set of the above set of clauses from the resolution, repeated here for convenience

$$[\neg P(x), P(fx)], [\neg P(x) P(ffx)], \ldots, [\neg P(x), P(f^{2^n}x)],$$
$$[P(a)], [P(f^{2^n}a)], [\neg P(f^{2^n}a)], \Box.$$

Since a is the only constant in our language $P_{n+5}=\{a\}$. Let $C_1, C_2, \ldots, C_{n+5}$ be the above refutation. Then θ_i, the unifying substitution for each C_i, is $\{y \mapsto f^{2^{i-1}}x\}$ for $2 \leq i \leq n+1$. The substitutions $\theta_1, \theta_{n+2}, \theta_{n+3}, \theta_{n+4}$, and θ_{n+5} corresponding to clauses $C_1, C_{n+2}, C_{n+3}, C_{n+4}$, and C_{n+5} are $\{\}, \{\}, \{x \mapsto a\}, \{\}$, and $\{\}$ respectively. So we obtain, by the calculation in the last section:

$$\begin{aligned}
P_{n+5} &= \{a\} \\
P_{n+4} &= \{a\} \\
P_{n+3} &= \{a\} \\
P_{n+2} &= \{a\} \\
P_{n+1} &= \{a\} \\
P_n &= \{a, f^{2^{n-1}}a\} \\
P_{n-1} &= \{a, f^{2^{n-2}}a, f^{2^{n-1}}a, f^{2^{n-1}+2^{n-2}}a\} \\
&\vdots \\
P_2 &= \{a, f^2a, f^4a, \ldots, f^{2^n-2}a\} \\
P_1 &= \{a, fa, f^2a, \ldots, f^{2^n-1}a\}.
\end{aligned}$$

And, this is exactly the proof set that was required in the previous ground resolution example. To see that P_1 is $\{f^k a | 0 \leq k \leq 2^n - 1\}$ observe that for $1 \leq k \leq n$ that $P_k = P_{k+1} \cup \{f^{n+2^{k-1}} a | f^n a \in P_k\}$. This doubles the number of elements in each successive set P_k because at each stage we are adding a distinct power of 2 to the exponent of f. There is one element in P_{n+1} and n doublings to get to P_1 so there must be 2^n elements in P_1. So P_1 is indeed the required proof set for this example.

Considering the improvement that resolution has over the previous methods of automated theorem proving one may legitimately ask why resolution isn't actually used by researchers to solve problems. While resolution is a dramatic improvement over the level-saturation methods that went before, it is generally held that resolution is also inadequate to solve the kinds of problems such people might want to present to it. The research that followed Robinson's result at first attempted to refine the resolution rule (the calculation of resolvants) while maintaining logical completeness. These improvements were still inadequate. Later research went in the direction of sacrificing the completeness of the method for computational efficiency. In particular the Prolog programming language has had some success. Now research on resolution-based automated theorem proving continues with these two disparate foci: slow but complete methods, and somewhat faster but incomplete methods.

Bibliography

[Baaz92] Baaz, M. and Leitsch, A., Complexity of resolution proofs and function introduction, Annals of Pure and Applied Logic, 57(1992), 181–215.

[Davis60] Davis, M. and Putnam, H., A computing procedure for quantification theory, Journal of the ACM, 7(1960), 201–215.

[Enderton72] Enderton, H., A Mathematical Introduction to Logic, Academic Press, Orlando, 1972.

[Fitting94] Fitting, Melvin, First-Order Logic and Automated Theorem Proving, Springer-Verlag, New York, 1994.

[Godel34] Gödel, K., On Undecidable Propositions of Formal Mathematical Systems, Institute for Advanced Study Report, 1934.

[Loveland79] Loveland, Donald W., Automated Theorem Proving: A Logical Basis, Elsevier North-Holland, Amsterdam, 1979.

[Paterson78] Paterson, M. S., and Wegman, M. N., Linear Unification, Journal of Computer and System Science 16(2), 158–167.

[Prawitz60] Prawitz, D., An improved proof procedure, Theoria 26(1960), 102–139.

[Robinson65] Robinson, J. A., A machine-oriented logic based on the resolution principle, Journal of the ACM, 1(1965), 23–41.

[Shoenfield67] Shoenfield, J., Mathematical Logic, Addison-Wesley, New York, 1967.

[Urquhart87] Urquhart, A., Hard examples for resolution, Journal of the ACM, 34(1987), 209–219.