A Deductive System for Real Algebra

Frank M. Brown
Department of Computer Sciences
The University of Texas at Austin
Austin, Texas

TR 141 March 1980

A Deductive System for Real Algebra

Abstract

We describe an automatic theorem prover for real algebra which is based on Symmetric Logic. Symmetric Logic unlike Sequent Logic, Natural Deduction and Resolution, is based on treating the universal and existential quantifiers analogously, and on treating conjunctions and disjunctions analogously. Proofs of various theorems of real algebra involving mathematical induction have been obtained.

Contents

1. Introduction

In the past two decades three basic types of logical inference systems have been proposed:

(1) Resolution systems such as Robinson [1]

(2) Natural Deduction systems such as Bledsoe [2,3], and Pastre [4,5]

and (3) Sequent Calculus Systems such as Brown [6,7,8,9,10,11,12,13], and Bibel [14].

However, these three basic types of systems are similar in that they are all based on representing logical sentences in some specific Skolem Prenix Normal Form [9] and performing deduction by use of a unification algorithm [1,9]. For example Sequent Logics and Natural deduction systems put theorems into Skolem Prenix Conjunctive Normal Form, and resolution (via duality) can be said to put theorems to prove into Skolem Prenix Disjunctive Normal Form.

Our experience with such systems [6,7,8,9,10,11,12,13] has convinced us that any system such as these which are based on the Skolem Prenix Normal Form and unification possess a number of inherent defects.

1.1. The Equality interface defect

One problem is that usch systems do not interface existential quantifiers in a goal to equality in the same efficient manner in which universal quantifiers are interfaced to equality. For example our sequent logic based number theory theorme prover [6,7] treats a universally quantified goal expression like $\rightarrow(\forall x \phi x\ \Psi x)$ as follows. First, the all sign is eliminated by the $\rightarrow \forall$ rule creating a Skolem constant "a".

$\rightarrow \phi a\ \Psi a$

Next the implication sign is eliminated with the $\rightarrow$ rule making ϕa an hypothesis: $\phi a \rightarrow \Psi a$

Next the system tries to rewrite ϕa as an equation of the form a=t

where "a" does not occur in "t": $a=t \rightarrow \Psi a$

Finally, a=t is eliminated by the $\Rightarrow$ rule replacing all occurances of a by t. The result of all this is to simplify the original goal without any search by decreasing its quantificational complexity in a manner which leaves a new subgoal ϕt to prove which is equivalent to and simpler than the original subgoal. For this reason the interaction of the $\rightarrow \forall$ and $\Rightarrow$ rules contribute significantly to the theorem provers ability to prove theorems containing universal quantifiers.

Consider now how existential quantifiers in a goal are treated by this number theory theorem prover: $\rightarrow \exists x(\phi x \wedge \Psi x)$

First the $\rightarrow \exists$ rule creates a copy of that goal containing a new unification variable: $*$

$\rightarrow \exists x(\phi x \wedge \Psi x), \phi_* \wedge \Psi_*$

The copy is then split by the $\rightarrow \wedge$ rule resulting in two sequents:

$\rightarrow \exists x(\phi x \wedge \Psi x), \phi_*$ $\rightarrow \exists x(\phi x \wedge \Psi x), \Psi_*$

resulting in essentially another copy of the original goal. An additional indeterminite number of further copies may be added by repeated applications of the $\rightarrow \exists$ and $\rightarrow \wedge$ rules.

Let us suppose now that $\phi*$ can be manipulated into the form $*=t$ where * does not occur in it.

$\rightarrow \exists x(\phi x \wedge \Psi x), {}_*=t$ $\rightarrow \exists x(\phi x \wedge \Psi x), \Psi_*$

Our Number Theory Theorem Prover would then try to unify these sequents by finding substitution instances for the unification variables which would make these sequents tantologous. For example if t were substituted for $*$ the first sequent would be tantologons leaving only the second sequent $\rightarrow \exists x \phi x \wedge \Psi x, \Psi t$

Although this looks rather efficient it isn't in a complex theorem in which many sequents and unification variables are produced, became in

this case the substitution t for $_*$, is only one of many possible substitutions which might solve a given sequent, and the system has no way of choosing the right one. Note also that this substitution does not make the second sequent tantologons and so might not even be among the possible substitutions. Thus we see that existential quantifiers are not interfaced to equality in an efficient manner.

1.2. The Conditional item interface defect

A second problem is that existential quantifiers are not interfaced to conditional items in a reasonable manner as are universal quantifiers.

Consider for example a conditional item relating to a sort C:

$Cx \rightarrow tx=sx$

which indicates that we want to replace tx by sx whenever x is of the right sort.

In the case of a universal quantifier our goal might be:

$\rightarrow \forall x(Cx \supset \phi(tx))$:

which by $\rightarrow \forall$ and then $\rightarrow \supset$ becomes,

$\rightarrow Ca \supset \phi(ta)$

$Ca \rightarrow \phi(ta)$

The item is now easily preformed by searching the hypotheses in that sequent for a sentence; namely Ca; which matches the Cx sentence of the conditional items. However, this check is not easily performed in the case of an existential quantifier in a goal:

$\rightarrow \exists x\ Cx \wedge \phi tx$

Using the $\rightarrow \exists$ and then the $\rightarrow \wedge$ rules once we get:

$$\rightarrow(\exists xCx \wedge \phi tx), C^* \wedge \phi t^*$$

and then:

$$\rightarrow(\exists xCx \wedge \phi tx), C^* \qquad \rightarrow(\exists xCx \wedge \phi tx), \phi t^*$$

Now, however it is practically impossible to retrieve the fact that $*$ in ϕt_* is in fact of sort C, and thus we cannot apply the conditional item to it.

1.3. Summary

We have shown that logical deduction systems based on the Skolem Prenix Conjunctive normal form have some inherent defects. Our experience with automatic theorem proving convinces us that for equational theories or theories with many sorts, that these defects will be crucial.

For this reason, we have begun to experiment with a new kind of logical system called Symmetric Logic which treats Universal and Existential quantifiers in strictly analogous ways. This new logic is described in section 2 of this paper, and is applied to proving theorems in the equation based theory of Real Algebra. Some example proofs obtained by this theorem prover are given in Section 3.

2. Description of the Theorem Prover

Our theorem prover consists of an interpreter for mathematical expressions and many items of mathematical knowledge. This interpreter is a fairly complex mechanism, but it may be viewed as applying active items of mathematical knowledge of the form: $\phi \leftrightarrow \psi$ or $\phi = \psi$ to the theorem being proven, in the following manner. The interpreter evaluates the theorem recursively in a call-by-need manner. That is, if $(a_1 \ldots a_n)$ is a sub-expression being evaluated, then the interpreter tries to apply its currently active items of knowledge to that sub-expression before evaluating the arguments $a_1 \ldots a_n$. For each sub-expression that the interpreter evaluates, in turn in tries to match the ϕ expression of an item tc that sub-expression. If, however, during the application process an argument a_i does not match the corresponding argument of the ϕ expression, then a_i is evaluated, and the system then tries to match the result of that evaluation. If ever the interpreter finds a sub-expression $\phi\theta$ which is an instance of ϕ of some active item, then it replaces that expression by the corresponding instance $\psi\theta$ or ψ. At this point all memory of the sub-expression $\phi\theta$ is immediately lost and the interpreter now evaluates $\psi\theta$. If no active items can be applied to a sub-expression then the sub-expression is not evaluated again but is simply returned.

For example, if

$x + z = y + z \leftrightarrow x = y$

and $x = x \leftrightarrow$ [redacted]

are the only items, and if they are listed to be used in that order, then evaluating the theorem

$y + z = y + z$

will initially cause it to be replaced by:

$y = y$

All memory of the subexpressions $y + z = y + z$ is immediately lost upon

its replacement by y = y and thus the interpreter does not attempt to apply the second item to y + z = y + z.

This interpreter has been used to prove theorems in several mathematical domains including number theory [6,7], set theory [8,9,10], meta mathematics [11,12], and Intensional Logic [13]. The items used by the interpreter in a particular domain are intended to be theorems of that domain. Thus for example, if Γ is a conjunction of axioms for a particular domain, and if $\phi \leftrightarrow \psi$ is an item used in that domain, then $\Gamma \rightarrow (\phi \leftrightarrow \psi)$ should be logically valid.

Sometimes it will be the case that our interpreter will need to use items which are valid only in certain sub-domains of a given domain. For example, in real algebra, if we wish to prove theorems involving natural numbers (note that a natural number is a type of real number) it would be quite useful to have available items which are valid only in the domain of natural numbers, or more precisely when certain free variables occuring in the item are restricted to being natural numbers. The representation of the item would be:

$$(\text{Nat } n) \rightarrow (\phi n \leftrightarrow \psi n)$$

More generally then, if we wish to use an item $\phi x \leftrightarrow \psi x$ (or $\phi x = \psi x$) where x is restricted to the subdomain Πx, then we represent it by a conditional item:

$$\Pi x \rightarrow (\phi x \leftrightarrow \psi x)$$

or $\Pi x \rightarrow (\phi x = \psi x)$

Note that $(\Gamma \wedge \Pi x) \rightarrow (\phi x \leftrightarrow \psi x)$

or $(\Gamma \wedge \Pi x) \rightarrow (\phi x = \psi x)$

should then be logically valid.

The interpreter handles conditional items in the same way in which it handles non-conditional items until it has found a $\phi\theta$ which matches the sub-expression being evaluated. At this point on a conditional item, the

interpreter tries to match each element in the conjunction Πx with some expression which it believes to be true. If such matches are found with substitution $\theta\sigma$ then $\psi\theta\sigma$ is returned. Otherwise the interpreter tries to apply another item as previously described.

The use of conditional items thus provides a general method of restricting the free variables of an item to a particular sub-domain. Its only disadvantage is that the amount of extra matching it forces the interpreter to perform. In order to minimize the amount of matching on the most common sub-domains we allow those sub-domains to be indicated by a particular style of variables.

At any given time an item may either be active or inactive. When an item is active it is at that time available for use in the evaluation process, but when it is inactive the interpreter simply ignores it. Items can be caused to be activated or deactivated by the interpreters use or even attempted use of other items. This in effect items cause the activation or deactivation, as the implementor specifies, of other items.

The purpose of allowing items to activate or deactivate other items is to allow a group of items to work toward achieving a common goal. For example in order to minimize the scope of quantifiers the subformula of a universal quantifier should be put into conjunctive normal form, whereas the sub-formula of an existential quantifier should be put in disjunctive normal form. To achieve this the evaluation of a universal quantifier activates the distributive law

$$((p \wedge q) \vee s) \leftrightarrow ((p \vee s) \quad (q \vee s))$$

and deactivates other distributive law:

$$((p \vee q) \wedge s) \leftrightarrow [(p \wedge s) \vee (q \wedge s)]$$

whereas the evaluation of the existential quantifier does just the reverse.

Tables of rewrite rules, rather simple items represented in ones mathematical language and other items were used in many of Bledsoe's theorem

provers [2,3]. Boyer and Moore [15] used a symbolic LISP interpreter to order the application of various recursive definitions, rewrite rules, and induction rules.

This Theorem prover uses both logical and Real Algebraic Knowledge. We first describe the items of logical knowledge in section 2.1 and the items for real algebra in section 2.2.

2.1. Symmetric Logic

Symmetric Logic is based on the idea that universal and existential quantifiers should be treated analogously, and that conjunction and disjunction should be treated analogously. Thus, instead of trying to manipulate goals in skolem prenix conjunctive normal form (The conjunction of sequents with each sequent being a disjunction) it manipulates quantifiers into miniscope form without any skolemization. The sentence argument of a universal quantifier is put into conjunctive normal form, and the sentence of an existential quantifier is put into disjuctive normal form so as to allow the quantifiers to pass thru that conjunction or disjunction using the Laws:

$$\forall x \quad \phi x \wedge \psi x \leftrightarrow \forall x \phi x \wedge \forall x \psi x$$

$$\exists x \quad \phi x \vee \psi x \leftrightarrow \exists x \phi x \vee \exists x \psi x$$

Since quantifiers are pushed to as low a scope as possible it follows that they will appear in front of either a disjunction or conjunction of either a literal or a quantified sentence:

$$\forall x \quad \sim\phi x \vee \psi x$$

$$\exists x \quad \phi x \wedge \psi x$$

Thus if for example ϕx is an equation $x = t$

$$\forall x \sim x = t \vee \psi x$$

$$\exists x \; x = t \wedge \psi x$$

we can now use the laws:

$(\forall x \sim x = t \rightarrow \psi x) \leftrightarrow \psi t$

$(\exists x\ x = t \wedge \psi x) \leftrightarrow \psi t$

to reduce the quantificational complexity of those goals. This solves the equality interface problem.

Suppose now that ϕx is a sort predicate Cx for some conditional item

$$Cx \rightarrow tx = sx$$

then when we see tx all we need do is search the disjunction $\sim \phi x \vee \psi x$ if x is universally quantified, or search the conjunction $\phi x \wedge \psi x$ if x is exisistentially quantified. Thus, Symmetric logic also solves the conditional item interface problem.

The syntax of symmetric logic consists of ten logical symbols which are listed below with their English translations:

$\wedge$	and
$\vee$	or
$\sim$	not
■	true
□	false
$\rightarrow$	implies
$\leftrightarrow$	iff
$\exists$	there exists
$\forall$	for all
$\vdash$	entails

The items for Symmetric Logic are listed below.

2.2.1. Conjunction Laws:

PCA1:	$\square \wedge p \leftrightarrow p$	Identity of conjunction
PCA2:	$\square \wedge p \leftrightarrow \square$	Zero of conjunction
PCA3:	$(p \vee q) \wedge s \leftrightarrow (p \vee s) \wedge (q \vee s)$	Distribution of conjunction over disjunction
PCA4:	$(p \wedge q) \wedge s \leftrightarrow p \wedge (q \wedge s)$	Assositivity of conjunction
PCA5:	$p \wedge \square \leftrightarrow \square$	Zero of conjunction
PCA6:	$p \wedge \square \leftrightarrow \square$	Identity of conjunction
PCA7:	$s \wedge (p \vee q) \leftrightarrow (s \vee p) \wedge (s \vee q)$	Distribution of conjunction over disjunction
PCA8:	$p \wedge \ldots \wedge p \wedge \ldots \leftrightarrow \ldots \wedge p \wedge \ldots$	Idempotency of conjunction
PCA9:	$\sim p \wedge \ldots \wedge p \wedge \ldots \leftrightarrow \square$	Inverse of conjunction
PCA10:	$p \wedge \ldots \wedge \sim p \wedge \ldots \leftrightarrow \square$	Inverse of conjunction
PCA11:	$x = t \wedge \ldots \wedge px \wedge \ldots \leftrightarrow x = t \wedge \ldots \wedge pt \wedge \ldots$ $Px \wedge \ldots \wedge x = t \wedge \ldots \leftrightarrow Pt \wedge \ldots \wedge x = t \wedge \ldots$	conjunctive equality

2.1.2. Disjunction Laws

PC01:	■ v p ↔ ■	Zero of disjunction
PC02:	□v p ↔ p	Identity of disjunction
PC03:	(p ∧q)vs ↔ (pvs) ∧(qvs)	distribution of disjunction over conjunction
PC04:	(pvq) v s ↔ pv(qvs)	associtivity of disjunction
PC05:	p v ■ ↔ ■	Zero of disjunction
PC06:	p v □ ↔ p	Identity of disjunction
PC07:	s v (p ∧q) ↔ (s v p) ∧(s v q)	distribution of disjunction
PC08:	p v ... v p v... ↔ ...v p v....	idempotency of disjunction
PC09:	p v ... v∿pv... ↔ ■	inverse of disjunction
PC010:	∿pv...vpv... ↔ ■	inverse of disjunction
PC011:	(∿x = t)v...vPx... ↔ (x = t) ... Ptv... Px v ... v(∿x = t)v... ↔ (Pt)v...v(x = t)	disjunctive equality

2.1.3. Negation Laws

PCN1: $\sim$[redacted] $\leftrightarrow$ [redacted] :Truth table

PCN2: $\sim$[redacted] $\leftrightarrow$ [redacted] :Truth table

PCN3: $\sim (p \wedge q) \leftrightarrow \sim p \vee \sim q$:DeMorgans Law

PCN4: $\sim (p \vee q) \leftrightarrow \sim P \wedge \sim q$:DeMorgans Law

PCN5: $\sim \forall x Px \leftrightarrow \exists x \sim Px$:Generalized DeMorgans Law

PCN6: $\sim \exists x Px \leftrightarrow \forall x \sim Px$:Generalized DeMorgans Law

PCN7: $\sim\sim p \leftrightarrow p$:Double negation

2.1.4. Universal Quantification Laws

PCAL1: $\forall x Px \wedge Qx \leftrightarrow \forall x Px \wedge \forall x Qx$

PCAL2: $\forall x p \leftrightarrow p$

PCAL3: $\forall x(\ldots vqv\ldots) \leftrightarrow q v \forall x(\ldots v\ldots)$

PCAL4: $\forall x(Pxvx \neq tvQx) \leftrightarrow (Ptvt \neq tvQt)$: Universal equality

PCAL1, PCAL2, PCAL3, PCAL4 all:

1. activate PCO3, PCO4, PCO10, (Basevar x)
2. deactivate PCA3, PCA7, PCA9, PC10, (Basevar x)

Whenever symeval quits trying to apply a PCAL law these PCO and PCA laws are restored to their previous activation states automatically.

2.1.5. Existential Quantification

PCEX1: $\exists x P x \vee Q x \leftrightarrow \exists x P x \vee \exists x Q x$

PCEX2: $\exists x p \leftrightarrow p$

PCEX3: $\exists x(\ldots \wedge q \wedge \ldots) \leftrightarrow q \wedge \exists x(\ldots \wedge \ldots)$

PCEX4: $\exists x(Px \wedge x{=}t \wedge Qx) \leftrightarrow (Pt \wedge t{=}t \wedge Qt)$: Existential equality

PCEX1, PCEX2, PCEX3, PCEX4 all:

1. activate PCA3, PCA7, PCA9, PCA10, (Basevar x)
2. deactivate PCO3, PCO7, PCO9, PCO10, (Basevar x)

Whenever symeval quits trying to apply a PCEX law these PCA and PCO laws are restored to their previous activation states automatically.

2.1.6. Implication Laws

PCIF1: $(p \supset q) \leftrightarrow (\sim p \vee q)$: definition

2.1.7. Bi Implication Laws

PCIFF1: $(\blacksquare \leftrightarrow p) \leftrightarrow p$: truth table

PCIFF2: $(\square \leftrightarrow p) \leftrightarrow \sim p$: truth table

PCIFF3: $(p \leftrightarrow \blacksquare) \leftrightarrow p$: truth table

PCIFF4: $(p \leftrightarrow \square) \leftrightarrow \sim p$: truth table

PCIFF5: $(p \leftrightarrow p) \leftrightarrow \blacksquare$: equivalence

PCIFF6: $(p \leftrightarrow q) \leftrightarrow ((\sim p \vee q) \wedge (\sim q \vee p))$: definition

2.1.8. Metalogic Laws

Three laws of our metalogic [11,12] are used.

V1: $\vdash_R \Box \leftrightarrow \Box$

V2: $\vdash_R \Box \leftrightarrow \Box$ assumes that the axioms of real algebra are consistent $\Diamond R$. (i.e. $\sim \vdash \sim R$)

V3: $\vdash_R (p \wedge q) \leftrightarrow \vdash_R p \wedge \vdash_R q$

The symbol $\vdash$ is interpreted as the modal concept of logical truth. R is the conjunction of all items in our theorem prover. Thus $\vdash_R p$ is interpreted as meaning that p follows from these items by the laws of logic. A more detailed description of this metalogic is given in [11,12,13,16].

2.2. Real Algebra

Our Real Algebra consists of the following algebraic symbols which are listed below with their English translations:

+	Plus
·	Times
-	Unary minus
-	Binary Subtraction
/.	division
↑	exponentiation
=	equality

The algebraic laws used by SYMEVAL are given in the succeeding subsections.

2.2.1. Plus Laws

RP1: $0 + x = x$

RP2: $x + 0 = x$

RP3: $(x + y) + z = x + (y + z)$

*RP4: $\overline{\overline{n}}\cdot(\overline{\overline{k}}\uparrow -1)\cdot x + \ldots + \overline{\overline{m}}\cdot(\overline{\overline{j}}\uparrow -1)\cdot x = \overline{\overline{\overline{\overline{n}}\cdot j + m\cdot K}}\cdot\overline{\overline{K\cdot j}\uparrow -1}\cdot x$

RP4 includes all lemma schemas obtained from the above by omitting sub-pieces of it.

For example: $\overline{\overline{n}}\cdot x + \ldots + \overline{\overline{m}}\cdot x = \overline{\overline{n + m}}\cdot x$

is inherent in RP4.

*note--a double bar is written over any expression which evaluates to a numeral.

2.2.2. Times Laws

RM1: $0 \cdot x = 0$

RM2: $x \cdot 0 = 0$

RM3: $1 \cdot x = x$

RM4: $x \cdot 1 = x$

RM5: $(x \cdot y) \cdot z = x \cdot (y \cdot z)$

RM6: $\overline{\overline{n}} \cdot \overline{\overline{m}} = \overline{\overline{n \cdot m}}$

RM7: $\overline{\overline{n}} \cdot (\overline{\overline{m}} \cdot x) = \overline{\overline{n \cdot m}} \cdot x$

RM8: $x \cdot \overline{\overline{n}} = \overline{\overline{n}} \cdot x$ (if x is not a numeral)

RM9: $x \cdot (\overline{\overline{n}} \cdot y) = \overline{\overline{n}} \cdot (x \cdot y)$ (if x is not a numeral)

*RM10: $v \uparrow \overline{\overline{n}} \cdot \ldots \cdot v \uparrow \overline{\overline{m}} = \ldots \cdot v \, \overline{\overline{m + n}}$

(if v is a variable, sign [m] = sign [n])

note: $v \cdot \ldots \cdot v \uparrow \overline{\overline{m}} \ldots = \ldots \cdot v \uparrow \overline{\overline{m + 1}} \ldots$

$v \uparrow \overline{\overline{n}} \cdot \ldots \cdot v \ \ldots = \ldots \cdot v \uparrow \overline{\overline{n + 1}}$

$v \cdot \ldots \cdot v \ldots = \ldots \cdot v \uparrow 2$

are all inherent in RM10.

RM11: $(-x \cdot y) = -(x \cdot y)$

RM12: $(x \cdot -y) = -(x \cdot y)$

RM13: $(x + y) \cdot z = x \cdot z + y \cdot z$

RM14: $z \cdot (x + y) = z \cdot x + z \cdot y$

} z is of the form $\overline{\overline{n}}$ or $\overline{\overline{n}} \uparrow -1$ or variable v is in x + y and the (BASEVAR v) strategy is active

*RM15: $\overline{\overline{n}} \cdot \ldots \cdot (\overline{\overline{m}} \uparrow -1) = \overline{\overline{n/GCD[nm]}} \cdot \ldots \cdot \overline{\overline{(m/GCD[nm]} - 1)}$

RM16: $\overline{\overline{n}} \uparrow -1 \cdot \overline{\overline{m}} \uparrow -1 = \overline{\overline{n \cdot m}} \uparrow -1$

RM17: $\overline{\overline{n}} \uparrow -1 \cdot (\overline{\overline{m}} \uparrow -1 \cdot x) = \overline{\overline{n \cdot m}} \uparrow -1 \cdot x$

RM18: $x \cdot \overline{\overline{n}} \uparrow -1 = \overline{\overline{n}} \uparrow -1 \cdot x$

RM19: $x \cdot (\overline{\overline{n}} \uparrow -1 \, y) = \overline{\overline{n}} \uparrow -1 \cdot (x \cdot y)$

} x is not of the form n or n -1

RM20: $v \cdot x = x \cdot v$

RM21: $v \cdot (x \cdot y) = x \cdot (v \cdot y)$

} v is not in x and (Basevar v) is active

2.2.3. Minus Laws

RMIN1: $-(-x) = x$

RMIN2: $-(x + y) = (-x) + (-y)$

RMIN3: $-\overline{\overline{n}} = -\overline{\overline{n}}$

RMIN4: $-(\overline{\overline{n}} \cdot x) = (-\overline{\overline{n}} \cdot x)$

2.2.4. Subtraction Laws

RSUB1: $x \dot{-} y = x + (-y)$

2.2.5. Division Laws

RQUO1: $x/y = x \cdot (y \uparrow -1)$

2.2.6. Exponentiation Laws

REXP1: $x\uparrow 0 = 1$

REXP2: $x\uparrow 1 = x$

REXP3: $1\uparrow x = 1$

REXP4: $0\uparrow\overline{\overline{n}} = 0$ if $(n\neq 0)$

REXP5: $(x\uparrow\overline{\overline{n}})\uparrow\overline{\overline{m}} = x\uparrow\overline{\overline{n\cdot m}}$ if (n,m are > o)

REXP6: $(x\cdot y)\uparrow\overline{\overline{n}} = (x\uparrow\overline{\overline{n}})\cdot(y\uparrow\overline{\overline{n}})$

REXP7: $(x + y)\uparrow\overline{\overline{n}} = \sum_{k=0}^{n} \binom{n}{k} \cdot (x\uparrow n) \cdot (y\uparrow(n-k))$

(Binomial Theorem) if $(n>o)$

REXP8: $\overline{\overline{n}}\uparrow\overline{\overline{m}} = \overline{\overline{n}}\cdot(\overline{\overline{n}}\uparrow(\overline{\overline{m-1}}))$ (if $n\neq 0$, $m>0$)

REXP9: $\overline{\overline{n}}\uparrow\overline{\overline{m}} = (\overline{\overline{n}}\uparrow-\overline{\overline{m}})\uparrow -1$ (if $n\neq 0, m<-1$)

REXP10: $\overline{\overline{n}}\uparrow -1 = (-(\overline{\overline{-n}}\uparrow -1))$ (if $n<0$)

REXP11: $\overline{\overline{n}}\uparrow(x + y) = (n\uparrow x)\cdot(n\uparrow y)$ (if $n\neq 0$)

2.2.7. Equality Laws

REQ1: $x = x \leftrightarrow$

REQ2: $\overline{\overline{n}} = \overline{\overline{m}} \leftrightarrow$ (if $n \neq m$)

REQ3: $(\overline{\overline{n}} \cdot x) = y \quad \leftrightarrow \quad x = ((\overline{\overline{n}}\uparrow -1) \cdot y)$

REQ4: $\underbrace{(+\ldots(\cdot\ldots x\uparrow-\overline{\overline{n}}\ldots)\ldots)}_{\alpha} = \underbrace{y}_{\beta} \leftrightarrow (x = 0 \vee x\uparrow\overline{\overline{n}} \cdot \alpha = x\uparrow\overline{\overline{n}} \cdot \beta) \wedge (x = 0 \rightarrow \alpha = \beta)$

$y = (+\ldots(\cdot\ldots x\uparrow-\overline{\overline{n}}\ldots)\ldots)$

REQ5: $\underbrace{(+(\cdot\overline{\overline{n}}_1 x_1)\ldots(\cdot\overline{\overline{n}}_k x))}_{\alpha} = \underbrace{(+(\cdot\overline{\overline{m}}_1 \cdot y_1)\ldots(\overline{\overline{m}}_J + y_J))}_{\beta}$

$\leftrightarrow \overline{\overline{GCD[mn]}}\uparrow -1 \cdot \alpha = \overline{\overline{GCD[mn]}}\uparrow -1 \cdot \beta$

(if $GCD[mn] \neq 1$)

(note: if n_i or n_j is omitted its treated as '1')

*REQ6: $(+\ldots x\ldots) = y \leftrightarrow (+\ldots\ \ldots) = (-x) + y$

(if (Basevar v) is active and $v \notin x$)

(Thus REQ6 is a Demon)

REQ7: $y = (+\ldots x\ldots) \leftrightarrow (-x) + y = (+\ldots\ \ldots)$

(if (Basevar v) is active and $v \in x$)

REQ8: $-v = x \leftrightarrow v = -x$ (v is a variable)

REQ9: $-1 \cdot v = x \leftrightarrow v = -x$ (v is a variable)

2.2.8. Miscellaneous Algebraic Laws

RAL1: $\forall x(+\ldots(c_1 \cdot f_x)\ldots(c_n \cdot f_x)\ldots) = (+\ldots(d_1 \cdot f_x)\ldots(d_m \cdot f_x)\ldots)$

$\leftrightarrow c_1 + \ldots + c_n = d_1 + \ldots + d_m \wedge \forall x(+\ldots\ \ldots\ \ldots) = (+\ldots\ \ldots\ \ldots)$

(if all the bases are independent and $c_1 \ldots c_n$, $d_1 \ldots d_m$ are all the coeficients with an f_x base)

$\forall x$: Rallinduct: $\forall n \phi n \leftrightarrow (\phi o \wedge \forall n(\phi n \rightarrow \phi n+1))$

This law activates fertilization.

=: Fertilize: $(x+y = z \rightarrow x+u = v) \leftrightarrow z-v+u = v$

Where P is the symbol being recursed upon in the induction step. Fertilize deactivates itself.

$\forall x$: Rallinit: **no item**

This law activates REQ3 , REQ6 , REQ7 , and (Basevar x) in order to try to solve for x.

$\exists x$: Rexinit: no item

This law activates REQ3 , REQ6 , REQ7 , and (Basevar x) in order to try to solve for x.

Σ: Sigma: $\begin{cases} i \leq n \rightarrow \sum_{k=i}^{n} f_k = \left(\sum_{k=i}^{n-1} f_k\right) + f_n \\ i > n \rightarrow \sum_{k=i}^{n} f_k = 0 \end{cases}$

( ): RCO: $\binom{x}{y} = \frac{x!}{y!\ (x-y)!}$

!: RFAC1: $(n + 1)! = (n + 1) \cdot n!$

!: RFAC2: $0! = 1$

3. Examples

This theorem prover has proved a number of interesting theorems in Real Algebra. In this section we present two such theorems: The first is R23 which states that for all natural numbers is the sum as k goes from zero to n of three to the Kth power equals $((n+1^3)-1)/2$:

$$\text{R23: } \vdash_R \{\forall n \sum_{k=0}^{n} 3^k = \frac{(n+1)^3-1}{2}\}$$

The second theorem is RI5 which states that there exist terms x, y, z, u, v, and w such that the following is true in real algebra: for all natural numbers n the sum as k goes from 0 to n of k to the fourth power equals the polynominal: $x \cdot n^5 + y \cdot n^4 + z \cdot n^3 + u \cdot n^2 + v \cdot n + w$:

$$\text{RI5: } \exists x \exists y \exists z \exists u \exists v \exists w \vdash_R (\sum_{k=1}^{n} k^4 = x \cdot n^5 + y \cdot n^4 + z \cdot n^3 + u \cdot n^2 + v \cdot n + w)$$

It should be realized that RI5 implies but is not obviously implied by:

$$\vdash_R \{\exists x \exists y \exists z \exists u \exists v \exists w \ \forall n \sum_{k=1}^{n} k^4 = x \cdot n^5 + y \cdot n^4 + z \cdot n^3 + u \cdot n^2 + u \cdot n + n\}$$

because the theory of real algebra is not a complete theory.

The difference between these two sentences is somewhat like the difference in the sentences: There is someone whom I know to be a spy: $\exists x \vdash k(\text{Spy } x)$ I know that there is a spy: $\vdash k \exists x(\text{Spy } x)$. If there is someone whom you know to be a spy then clearly you know that there is a spy but the converse is rarely true.

3.1 The Proof of R23

The proof of R23 is presented in full detail as a linear list of input and output formulas labeled as In: or On: respectively. The natural number n in each label refers to the recursion level within the theorem prover. Thus each In: label is followed by an On: label which contains on its line a formula which is equivalent to the formula on the line containing that input label. The theorem prover produces the output line by applying one or more items to the formula on the input line and possibly evaluating subformulas of the formulae recursively.

```
TYPE R23

I:1 (ALL N (EQUAL (SIGMA K (: (0) N (EXP (3) K))) (QUO (SUB (EXP (3) (PLUS N (1)
)) (1)) (2))))
I:2 (EQUAL (SIGMA K (: (0) N (EXP (3) K))) (QUO (SUB (EXP (3) (PLUS N (1))) (1))
 (2)))
I:3 (SIGMA K (: (0) N (EXP (3) K)))
O:3 (SIGMA K (: (0) N (EXP (3) K)))
I:3 (QUO (SUB (EXP (3) (PLUS N (1))) (1)) (2))
O:3 (TIMES (SUB (EXP (3) (PLUS N (1))) (1)) (EXP (2) (-1)))
I:3 (TIMES (SUB (EXP (3) (PLUS N (1))) (1)) (EXP (2) (-1)))
I:4 (SUB (EXP (3) (PLUS N (1))) (1))
O:4 (PLUS (EXP (3) (PLUS N (1))) (MINUS (1)))
I:4 (PLUS (EXP (3) (PLUS N (1))) (MINUS (1)))
I:5 (EXP (3) (PLUS N (1)))
I:6 (PLUS N (1))
O:6 (PLUS N (1))
O:5 (TIMES (EXP (3) N) (EXP (3) (1)))
I:5 (TIMES (EXP (3) N) (EXP (3) (1)))
I:6 (EXP (3) N)
O:6 (EXP (3) N)
I:6 (EXP (3) (1))
O:6 (3)
O:5 (TIMES (3) (EXP (3) N))
I:5 (TIMES (3) (EXP (3) N))
I:6 (EXP (3) N)
O:6 (EXP (3) N)
O:5 (TIMES (3) (EXP (3) N))
I:5 (MINUS (1))
O:5 (-1)
O:4 (PLUS (TIMES (3) (EXP (3) N)) (-1))
I:4 (PLUS (TIMES (3) (EXP (3) N)) (-1))
I:5 (TIMES (3) (EXP (3) N))
I:6 (EXP (3) N)
O:6 (EXP (3) N)
O:5 (TIMES (3) (EXP (3) N))
O:4 (PLUS (TIMES (3) (EXP (3) N)) (-1))
I:4 (EXP (2) (-1))
O:4 (EXP (2) (-1))
O:3 (TIMES (EXP (2) (-1)) (PLUS (TIMES (3) (EXP (3) N)) (-1)))
I:3 (TIMES (EXP (2) (-1)) (PLUS (TIMES (3) (EXP (3) N)) (-1)))
I:4 (EXP (2) (-1))
O:4 (EXP (2) (-1))
I:4 (PLUS (TIMES (3) (EXP (3) N)) (-1))
I:5 (TIMES (3) (EXP (3) N))
I:6 (EXP (3) N)
O:6 (EXP (3) N)
O:5 (TIMES (3) (EXP (3) N))
O:4 (PLUS (TIMES (3) (EXP (3) N)) (-1))
O:3 (PLUS (TIMES (EXP (2) (-1)) (TIMES (3) (EXP (3) N))) (TIMES (EXP (2) (-1)) (
-1)))
I:3 (PLUS (TIMES (EXP (2) (-1)) (TIMES (3) (EXP (3) N))) (TIMES (EXP (2) (-1)) (
-1)))
I:4 (TIMES (EXP (2) (-1)) (TIMES (3) (EXP (3) N)))
I:5 (EXP (2) (-1))
O:5 (EXP (2) (-1))
I:5 (TIMES (3) (EXP (3) N))
I:6 (EXP (3) N)
O:6 (EXP (3) N)
O:5 (TIMES (3) (EXP (3) N))
O:4 (TIMES (3) (TIMES (EXP (2) (-1)) (EXP (3) N)))
I:4 (TIMES (3) (TIMES (EXP (2) (-1)) (EXP (3) N)))
```

```
I:5 (TIMES (EXP (2) (-1)) (EXP (3) N))
I:6 (EXP (2) (-1))
O:6 (EXP (2) (-1))
I:6 (EXP (3) N)
O:6 (EXP (3) N)
O:5 (TIMES (EXP (2) (-1)) (EXP (3) N))
O:4 (TIMES (3) (TIMES (EXP (2) (-1)) (EXP (3) N)))
I:4 (TIMES (EXP (2) (-1)) (-1))
I:5 (EXP (2) (-1))
O:5 (EXP (2) (-1))
O:4 (TIMES (-1) (EXP (2) (-1)))
I:4 (TIMES (-1) (EXP (2) (-1)))
I:5 (EXP (2) (-1))
O:5 (EXP (2) (-1))
O:4 (TIMES (-1) (EXP (2) (-1)))
O:3 (PLUS (TIMES (3) (TIMES (EXP (2) (-1)) (EXP (3) N))) (TIMES (-1) (EXP (2) (-
1))))
I:3 (PLUS (TIMES (3) (TIMES (EXP (2) (-1)) (EXP (3) N))) (TIMES (-1) (EXP (2) (-
1))))
I:4 (TIMES (3) (TIMES (EXP (2) (-1)) (EXP (3) N)))
I:5 (TIMES (EXP (2) (-1)) (EXP (3) N))
I:6 (EXP (2) (-1))
O:6 (EXP (2) (-1))
I:6 (EXP (3) N)
O:6 (EXP (3) N)
O:5 (TIMES (EXP (2) (-1)) (EXP (3) N))
O:4 (TIMES (3) (TIMES (EXP (2) (-1)) (EXP (3) N)))
I:4 (TIMES (-1) (EXP (2) (-1)))
I:5 (EXP (2) (-1))
O:5 (EXP (2) (-1))
O:4 (TIMES (-1) (EXP (2) (-1)))
O:3 (PLUS (TIMES (3) (TIMES (EXP (2) (-1)) (EXP (3) N))) (TIMES (-1) (EXP (2) (-
1))))
O:2 (EQUAL (SIGMA K (: (0) N (EXP (3) K))) (PLUS (TIMES (3) (TIMES (EXP (2) (-1)
) (EXP (3) N))) (TIMES (-1) (EXP^Z
 (2) (-1)))))
I:2 (EQUAL (SIGMA K (: (0) N (EXP (3) K))) (PLUS (TIMES (3) (TIMES (EXP (2) (-1)
) (EXP (3) N))) (TIMES (-1) (EXP^Z
 (2) (-1)))))
I:3 (SIGMA K (: (0) N (EXP (3) K)))
O:3 (SIGMA K (: (0) N (EXP (3) K)))
I:3 (PLUS (TIMES (3) (TIMES (EXP (2) (-1)) (EXP (3) N))) (TIMES (-1) (EXP (2) (-
1))))
I:4 (TIMES (3) (TIMES (EXP (2) (-1)) (EXP (3) N)))
I:5 (TIMES (EXP (2) (-1)) (EXP (3) N))
I:6 (EXP (2) (-1))
O:6 (EXP (2) (-1))
I:6 (EXP (3) N)
O:6 (EXP (3) N)
O:5 (TIMES (EXP (2) (-1)) (EXP (3) N))
O:4 (TIMES (3) (TIMES (EXP (2) (-1)) (EXP (3) N)))
I:4 (TIMES (-1) (EXP (2) (-1)))
I:5 (EXP (2) (-1))
O:5 (EXP (2) (-1))
O:4 (TIMES (-1) (EXP (2) (-1)))
O:3 (PLUS (TIMES (3) (TIMES (EXP (2) (-1)) (EXP (3) N))) (TIMES (-1) (EXP (2) (-
1))))
O:2 (EQUAL (SIGMA K (: (0) N (EXP (3) K))) (PLUS (TIMES (3) (TIMES (EXP (2) (-1)
) (EXP (3) N))) (TIMES (-1) (EXP^Z
 (2) (-1)))))
O:1 (AND (EQUAL (SIGMA K (: (0) (0) (EXP (3) K))) (PLUS (TIMES (3) (TIMES (EXP (
2) (-1)) (EXP (3) (0))))) (TIMES ^Z
```

```
(-1) (EXP (2) (-1))))) (ALL N (IF (EQUAL (SIGMA K (: (0) N (EXP (3) K))) (PLUS (
TIMES (3) (TIMES (EXP (2) (-1)) ^Z
(EXP (3) N))) (TIMES (-1) (EXP (2) (-1))))) (EQUAL (SIGMA K (: (0) (PLUS N (1))
(EXP (3) K))) (PLUS (TIMES (3) (^Z
TIMES (EXP (2) (-1)) (EXP (3) (PLUS N (1))))) (TIMES (-1) (EXP (2) (-1)))))))))
I:1 (AND (EQUAL (SIGMA K (: (0) (0) (EXP (3) K))) (PLUS (TIMES (3) (TIMES (EXP (
2) (-1)) (EXP (3) (0)))) (TIMES ^Z
(-1) (EXP (2) (-1))))) (ALL N (IF (EQUAL (SIGMA K (: (0) N (EXP (3) K))) (PLUS (
TIMES (3) (TIMES (EXP (2) (-1)) ^Z
(EXP (3) N))) (TIMES (-1) (EXP (2) (-1))))) (EQUAL (SIGMA K (: (0) (PLUS N (1))
(EXP (3) K))) (PLUS (TIMES (3) (^Z
TIMES (EXP (2) (-1)) (EXP (3) (PLUS N (1))))) (TIMES (-1) (EXP (2) (-1)))))))))
I:2 (EQUAL (SIGMA K (: (0) (0) (EXP (3) K))) (PLUS (TIMES (3) (TIMES (EXP (2) (-
1)) (EXP (3) (0)))) (TIMES (-1) ^Z
(EXP (2) (-1)))))
I:3 (SIGMA K (: (0) (0) (EXP (3) K)))
I:4 (SUB (0) (1))
O:4 (PLUS (0) (MINUS (1)))
I:4 (PLUS (0) (MINUS (1)))
O:4 (MINUS (1))
I:4 (MINUS (1))
O:4 (-1)
O:3 (PLUS (SIGMA K (: (0) (-1) (EXP (3) K))) (EXP (3) (0)))
I:3 (PLUS (SIGMA K (: (0) (-1) (EXP (3) K))) (EXP (3) (0)))
I:4 (SIGMA K (: (0) (-1) (EXP (3) K)))
O:4 (0)
O:3 (EXP (3) (0))
I:3 (EXP (3) (0))
O:3 (1)
I:3 (PLUS (TIMES (3) (TIMES (EXP (2) (-1)) (EXP (3) (0)))) (TIMES (-1) (EXP (2)
(-1))))
I:4 (TIMES (3) (TIMES (EXP (2) (-1)) (EXP (3) (0))))
I:5 (TIMES (EXP (2) (-1)) (EXP (3) (0)))
I:6 (EXP (2) (-1))
O:6 (EXP (2) (-1))
I:6 (EXP (3) (0))
O:6 (1)
O:5 (EXP (2) (-1))
I:5 (EXP (2) (-1))
O:5 (EXP (2) (-1))
O:4 (TIMES (3) (EXP (2) (-1)))
I:4 (TIMES (3) (EXP (2) (-1)))
I:5 (EXP (2) (-1))
O:5 (EXP (2) (-1))
O:4 (TIMES (3) (EXP (2) (-1)))
I:4 (TIMES (-1) (EXP (2) (-1)))
I:5 (EXP (2) (-1))
O:5 (EXP (2) (-1))
O:4 (TIMES (-1) (EXP (2) (-1)))
O:3 (PLUS (TIMES (4) (TIMES (EXP (4) (-1)) (1))) (0))
I:3 (PLUS (TIMES (4) (TIMES (EXP (4) (-1)) (1))) (0))
O:3 (TIMES (4) (TIMES (EXP (4) (-1)) (1)))
I:3 (TIMES (4) (TIMES (EXP (4) (-1)) (1)))
I:4 (TIMES (EXP (4) (-1)) (1))
O:4 (EXP (4) (-1))
I:4 (EXP (4) (-1))
O:4 (EXP (4) (-1))
O:3 (TIMES (1) (EXP (1) (-1)))
I:3 (TIMES (1) (EXP (1) (-1)))
O:3 (EXP (1) (-1))
I:3 (EXP (1) (-1))
O:3 (1)
```

```
O:2 (T)
O:1 (ALL N (IF (EQUAL (SIGMA K (: (0) N (EXP (3) K))) (PLUS (TIMES (3) (TIMES (E
XP (2) (-1)) (EXP (3) N))) (TIME^Z
S (-1) (EXP (2) (-1))))) (EQUAL (SIGMA K (: (0) (PLUS N (1)) (EXP (3) K))) (PLUS
 (TIMES (3) (TIMES (EXP (2) (-1)^Z
) (EXP (3) (PLUS N (1))))) (TIMES (-1) (EXP (2) (-1)))))))
I:1 (ALL N (IF (EQUAL (SIGMA K (: (0) N (EXP (3) K))) (PLUS (TIMES (3) (TIMES (E
XP (2) (-1)) (EXP (3) N))) (TIME^Z
S (-1) (EXP (2) (-1))))) (EQUAL (SIGMA K (: (0) (PLUS N (1)) (EXP (3) K))) (PLUS
 (TIMES (3) (TIMES (EXP (2) (-1)^Z
) (EXP (3) (PLUS N (1))))) (TIMES (-1) (EXP (2) (-1)))))))
I:2 (IF (EQUAL (SIGMA K (: (0) N (EXP (3) K))) (PLUS (TIMES (3) (TIMES (EXP (2)
(-1)) (EXP (3) N))) (TIMES (-1) ^Z
(EXP (2) (-1))))) (EQUAL (SIGMA K (: (0) (PLUS N (1)) (EXP (3) K))) (PLUS (TIMES
 (3) (TIMES (EXP (2) (-1)) (EXP ^Z
(3) (PLUS N (1))))) (TIMES (-1) (EXP (2) (-1))))))
O:2 (OR (NOT (EQUAL (SIGMA K (: (0) N (EXP (3) K))) (PLUS (TIMES (3) (TIMES (EXP
 (2) (-1)) (EXP (3) N))) (TIMES ^Z
(-1) (EXP (2) (-1)))))) (EQUAL (SIGMA K (: (0) (PLUS N (1)) (EXP (3) K))) (PLUS
(TIMES (3) (TIMES (EXP (2) (-1))^Z
 (EXP (3) (PLUS N (1))))) (TIMES (-1) (EXP (2) (-1))))))
I:2 (OR (NOT (EQUAL (SIGMA K (: (0) N (EXP (3) K))) (PLUS (TIMES (3) (TIMES (EXP
 (2) (-1)) (EXP (3) N))) (TIMES ^Z
(-1) (EXP (2) (-1)))))) (EQUAL (SIGMA K (: (0) (PLUS N (1)) (EXP (3) K))) (PLUS
(TIMES (3) (TIMES (EXP (2) (-1))^Z
 (EXP (3) (PLUS N (1))))) (TIMES (-1) (EXP (2) (-1))))))
I:3 (NOT (EQUAL (SIGMA K (: (0) N (EXP (3) K))) (PLUS (TIMES (3) (TIMES (EXP (2)
 (-1)) (EXP (3) N))) (TIMES (-1)^Z
 (EXP (2) (-1)))))))
I:4 (EQUAL (SIGMA K (: (0) N (EXP (3) K))) (PLUS (TIMES (3) (TIMES (EXP (2) (-1)
) (EXP (3) N))) (TIMES (-1) (EXP^Z
 (2) (-1)))))
I:5 (SIGMA K (: (0) N (EXP (3) K)))
O:5 (SIGMA K (: (0) N (EXP (3) K)))
I:5 (PLUS (TIMES (3) (TIMES (EXP (2) (-1)) (EXP (3) N))) (TIMES (-1) (EXP (2) (-
1))))
I:6 (TIMES (3) (TIMES (EXP (2) (-1)) (EXP (3) N)))
I:7 (TIMES (EXP (2) (-1)) (EXP (3) N))
I:10 (EXP (2) (-1))
O:10 (EXP (2) (-1))
I:10 (EXP (3) N)
O:10 (EXP (3) N)
O:7 (TIMES (EXP (2) (-1)) (EXP (3) N))
O:6 (TIMES (3) (TIMES (EXP (2) (-1)) (EXP (3) N)))
I:6 (TIMES (-1) (EXP (2) (-1)))
I:7 (EXP (2) (-1))
O:7 (EXP (2) (-1))
O:6 (TIMES (-1) (EXP (2) (-1)))
O:5 (PLUS (TIMES (3) (TIMES (EXP (2) (-1)) (EXP (3) N))) (TIMES (-1) (EXP (2) (-
1))))
O:4 (EQUAL (PLUS (MINUS (TIMES (3) (TIMES (EXP (2) (-1)) (EXP (3) N)))) (SIGMA K
 (: (0) N (EXP (3) K)))) (TIMES ^Z
(-1) (EXP (2) (-1))))
I:4 (EQUAL (PLUS (MINUS (TIMES (3) (TIMES (EXP (2) (-1)) (EXP (3) N)))) (SIGMA K
 (: (0) N (EXP (3) K)))) (TIMES ^Z
(-1) (EXP (2) (-1))))
I:5 (PLUS (MINUS (TIMES (3) (TIMES (EXP (2) (-1)) (EXP (3) N)))) (SIGMA K (: (0)
 N (EXP (3) K))))
I:6 (MINUS (TIMES (3) (TIMES (EXP (2) (-1)) (EXP (3) N))))
O:6 (TIMES (-3) (TIMES (EXP (2) (-1)) (EXP (3) N)))
I:6 (TIMES (-3) (TIMES (EXP (2) (-1)) (EXP (3) N)))
I:7 (TIMES (EXP (2) (-1)) (EXP (3) N))
```

```
I:10 (EXP (2) (-1))
O:10 (EXP (2) (-1))
I:10 (EXP (3) N)
O:10 (EXP (3) N)
O:7 (TIMES (EXP (2) (-1)) (EXP (3) N))
O:6 (TIMES (-3) (TIMES (EXP (2) (-1)) (EXP (3) N)))
I:6 (SIGMA K (: (0) N (EXP (3) K)))
O:6 (SIGMA K (: (0) N (EXP (3) K)))
O:5 (PLUS (TIMES (-3) (TIMES (EXP (2) (-1)) (EXP (3) N))) (SIGMA K (: (0) N (EXP
 (3) K))))
I:5 (PLUS (TIMES (-3) (TIMES (EXP (2) (-1)) (EXP (3) N))) (SIGMA K (: (0) N (EXP
 (3) K))))
I:6 (TIMES (-3) (TIMES (EXP (2) (-1)) (EXP (3) N)))
I:7 (TIMES (EXP (2) (-1)) (EXP (3) N))
I:10 (EXP (2) (-1))
O:10 (EXP (2) (-1))
I:10 (EXP (3) N)
O:10 (EXP (3) N)
O:7 (TIMES (EXP (2) (-1)) (EXP (3) N))
O:6 (TIMES (-3) (TIMES (EXP (2) (-1)) (EXP (3) N)))
I:6 (SIGMA K (: (0) N (EXP (3) K)))
O:6 (SIGMA K (: (0) N (EXP (3) K)))
O:5 (PLUS (TIMES (-3) (TIMES (EXP (2) (-1)) (EXP (3) N))) (SIGMA K (: (0) N (EXP
 (3) K))))
I:5 (TIMES (-1) (EXP (2) (-1)))
I:6 (EXP (2) (-1))
O:6 (EXP (2) (-1))
O:5 (TIMES (-1) (EXP (2) (-1)))
O:4 (AND (OR (EQUAL (2) (0)) (EQUAL (TIMES (EXP (2) (1)) (PLUS (TIMES (-3) (TIME
S (EXP (2) (-1)) (EXP (3) N))) (^Z
SIGMA K (: (0) N (EXP (3) K))))) (TIMES (EXP (2) (1)) (TIMES (-1) (EXP (2) (-1))
)))) (IF (EQUAL (2) (0)) (EQUAL ^Z
(PLUS (TIMES (-3) (TIMES (EXP (2) (-1)) (EXP (3) N))) (SIGMA K (: (0) N (EXP (3)
 K)))) (TIMES (-1) (EXP (2) (-1)^Z
)))))
I:4 (AND (OR (EQUAL (2) (0)) (EQUAL (TIMES (EXP (2) (1)) (PLUS (TIMES (-3) (TIME
S (EXP (2) (-1)) (EXP (3) N))) (^Z
SIGMA K (: (0) N (EXP (3) K))))) (TIMES (EXP (2) (1)) (TIMES (-1) (EXP (2) (-1))
)))) (IF (EQUAL (2) (0)) (EQUAL ^Z
(PLUS (TIMES (-3) (TIMES (EXP (2) (-1)) (EXP (3) N))) (SIGMA K (: (0) N (EXP (3)
 K)))) (TIMES (-1) (EXP (2) (-1)^Z
)))))
I:5 (OR (EQUAL (2) (0)) (EQUAL (TIMES (EXP (2) (1)) (PLUS (TIMES (-3) (TIMES (EX
P (2) (-1)) (EXP (3) N))) (SIGMA^Z
 K (: (0) N (EXP (3) K))))) (TIMES (EXP (2) (1)) (TIMES (-1) (EXP (2) (-1))))))
I:6 (EQUAL (2) (0))
O:6 (F)
O:5 (EQUAL (TIMES (EXP (2) (1)) (PLUS (TIMES (-3) (TIMES (EXP (2) (-1)) (EXP (3)
 N))) (SIGMA K (: (0) N (EXP (3)^Z
 K))))) (TIMES (EXP (2) (1)) (TIMES (-1) (EXP (2) (-1)))))
I:5 (EQUAL (TIMES (EXP (2) (1)) (PLUS (TIMES (-3) (TIMES (EXP (2) (-1)) (EXP (3)
 N))) (SIGMA K (: (0) N (EXP (3)^Z
 K))))) (TIMES (EXP (2) (1)) (TIMES (-1) (EXP (2) (-1)))))
I:6 (TIMES (EXP (2) (1)) (PLUS (TIMES (-3) (TIMES (EXP (2) (-1)) (EXP (3) N))) (
SIGMA K (: (0) N (EXP (3) K)))))
I:7 (EXP (2) (1))
O:7 (2)
I:7 (PLUS (TIMES (-3) (TIMES (EXP (2) (-1)) (EXP (3) N))) (SIGMA K (: (0) N (EXP
 (3) K))))
I:10 (TIMES (-3) (TIMES (EXP (2) (-1)) (EXP (3) N)))
I:11 (TIMES (EXP (2) (-1)) (EXP (3) N))
I:12 (EXP (2) (-1))
```

```
O:12 (EXP (2) (-1))
I:12 (EXP (3) N)
O:12 (EXP (3) N)
O:11 (TIMES (EXP (2) (-1)) (EXP (3) N))
O:10 (TIMES (-3) (TIMES (EXP (2) (-1)) (EXP (3) N)))
I:10 (SIGMA K (: (0) N (EXP (3) K)))
O:10 (SIGMA K (: (0) N (EXP (3) K)))
O:7 (PLUS (TIMES (-3) (TIMES (EXP (2) (-1)) (EXP (3) N))) (SIGMA K (: (0) N (EXP
 (3) K))))
O:6 (PLUS (TIMES (2) (TIMES (-3) (TIMES (EXP (2) (-1)) (EXP (3) N)))) (TIMES (2)
 (SIGMA K (: (0) N (EXP (3) K))))^Z
))
I:6 (PLUS (TIMES (2) (TIMES (-3) (TIMES (EXP (2) (-1)) (EXP (3) N)))) (TIMES (2)
 (SIGMA K (: (0) N (EXP (3) K))))^Z
))
I:7 (TIMES (2) (TIMES (-3) (TIMES (EXP (2) (-1)) (EXP (3) N))))
I:10 (TIMES (-3) (TIMES (EXP (2) (-1)) (EXP (3) N)))
I:11 (TIMES (EXP (2) (-1)) (EXP (3) N))
I:12 (EXP (2) (-1))
O:12 (EXP (2) (-1))
I:12 (EXP (3) N)
O:12 (EXP (3) N)
O:11 (TIMES (EXP (2) (-1)) (EXP (3) N))
O:10 (TIMES (-3) (TIMES (EXP (2) (-1)) (EXP (3) N)))
O:7 (TIMES (-6) (TIMES (EXP (2) (-1)) (EXP (3) N)))
I:7 (TIMES (-6) (TIMES (EXP (2) (-1)) (EXP (3) N)))
I:10 (TIMES (EXP (2) (-1)) (EXP (3) N))
I:11 (EXP (2) (-1))
O:11 (EXP (2) (-1))
I:11 (EXP (3) N)
O:11 (EXP (3) N)
O:10 (TIMES (EXP (2) (-1)) (EXP (3) N))
O:7 (TIMES (-3) (TIMES (EXP (1) (-1)) (EXP (3) N)))
I:7 (TIMES (-3) (TIMES (EXP (1) (-1)) (EXP (3) N)))
I:10 (TIMES (EXP (1) (-1)) (EXP (3) N))
I:11 (EXP (1) (-1))
O:11 (1)
O:10 (EXP (3) N)
I:10 (EXP (3) N)
O:10 (EXP (3) N)
O:7 (TIMES (-3) (EXP (3) N))
I:7 (TIMES (-3) (EXP (3) N))
I:10 (EXP (3) N)
O:10 (EXP (3) N)
O:7 (TIMES (-3) (EXP (3) N))
I:7 (TIMES (2) (SIGMA K (: (0) N (EXP (3) K))))
I:10 (SIGMA K (: (0) N (EXP (3) K)))
O:10 (SIGMA K (: (0) N (EXP (3) K)))
O:7 (TIMES (2) (SIGMA K (: (0) N (EXP (3) K))))
O:6 (PLUS (TIMES (-3) (EXP (3) N)) (TIMES (2) (SIGMA K (: (0) N (EXP (3) K)))))
I:6 (PLUS (TIMES (-3) (EXP (3) N)) (TIMES (2) (SIGMA K (: (0) N (EXP (3) K)))))
I:7 (TIMES (-3) (EXP (3) N))
I:10 (EXP (3) N)
O:10 (EXP (3) N)
O:7 (TIMES (-3) (EXP (3) N))
I:7 (TIMES (2) (SIGMA K (: (0) N (EXP (3) K))))
I:10 (SIGMA K (: (0) N (EXP (3) K)))
O:10 (SIGMA K (: (0) N (EXP (3) K)))
O:7 (TIMES (2) (SIGMA K (: (0) N (EXP (3) K))))
O:6 (PLUS (TIMES (-3) (EXP (3) N)) (TIMES (2) (SIGMA K (: (0) N (EXP (3) K)))))
I:6 (TIMES (EXP (2) (1)) (TIMES (-1) (EXP (2) (-1))))
I:7 (EXP (2) (1))
```

```
O:7 (2)
I:7 (TIMES (-1) (EXP (2) (-1)))
I:10 (EXP (2) (-1))
O:10 (EXP (2) (-1))
O:7 (TIMES (-1) (EXP (2) (-1)))
O:6 (TIMES (-2) (EXP (2) (-1)))
I:6 (TIMES (-2) (EXP (2) (-1)))
I:7 (EXP (2) (-1))
O:7 (EXP (2) (-1))
O:6 (TIMES (-1) (EXP (1) (-1)))
I:6 (TIMES (-1) (EXP (1) (-1)))
I:7 (EXP (1) (-1))
O:7 (1)
O:6 (-1)
O:5 (EQUAL (PLUS (TIMES (-3) (EXP (3) N)) (TIMES (2) (SIGMA K (: (0) N (EXP (3)
K))))) (-1))
I:5 (EQUAL (PLUS (TIMES (-3) (EXP (3) N)) (TIMES (2) (SIGMA K (: (0) N (EXP (3)
K))))) (-1))
I:6 (PLUS (TIMES (-3) (EXP (3) N)) (TIMES (2) (SIGMA K (: (0) N (EXP (3) K)))))
I:7 (TIMES (-3) (EXP (3) N))
I:10 (EXP (3) N)
O:10 (EXP (3) N)
O:7 (TIMES (-3) (EXP (3) N))
I:7 (TIMES (2) (SIGMA K (: (0) N (EXP (3) K))))
I:10 (SIGMA K (: (0) N (EXP (3) K)))
O:10 (SIGMA K (: (0) N (EXP (3) K)))
O:7 (TIMES (2) (SIGMA K (: (0) N (EXP (3) K))))
O:6 (PLUS (TIMES (-3) (EXP (3) N)) (TIMES (2) (SIGMA K (: (0) N (EXP (3) K)))))
O:5 (EQUAL (PLUS (TIMES (-3) (EXP (3) N)) (TIMES (2) (SIGMA K (: (0) N (EXP (3)
K))))) (-1))
I:5 (IF (EQUAL (2) (0)) (EQUAL (PLUS (TIMES (-3) (TIMES (EXP (2) (-1)) (EXP (3)
N))) (SIGMA K (: (0) N (EXP (3) ^Z
K)))) (TIMES (-1) (EXP (2) (-1))))))
O:5 (OR (NOT (EQUAL (2) (0))) (EQUAL (PLUS (TIMES (-3) (TIMES (EXP (2) (-1)) (EX
P (3) N))) (SIGMA K (: (0) N (EX^Z
P (3) K)))) (TIMES (-1) (EXP (2) (-1))))))
I:5 (OR (NOT (EQUAL (2) (0))) (EQUAL (PLUS (TIMES (-3) (TIMES (EXP (2) (-1)) (EX
P (3) N))) (SIGMA K (: (0) N (EX^Z
P (3) K)))) (TIMES (-1) (EXP (2) (-1))))))
I:6 (NOT (EQUAL (2) (0)))
I:7 (EQUAL (2) (0))
O:7 (F)
O:6 (T)
O:5 (T)
O:4 (EQUAL (PLUS (TIMES (-3) (EXP (3) N)) (TIMES (2) (SIGMA K (: (0) N (EXP (3)
K))))) (-1))
I:4 (EQUAL (PLUS (TIMES (-3) (EXP (3) N)) (TIMES (2) (SIGMA K (: (0) N (EXP (3)
K))))) (-1))
I:5 (PLUS (TIMES (-3) (EXP (3) N)) (TIMES (2) (SIGMA K (: (0) N (EXP (3) K)))))
I:6 (TIMES (-3) (EXP (3) N))
I:7 (EXP (3) N)
O:7 (EXP (3) N)
O:6 (TIMES (-3) (EXP (3) N))
I:6 (TIMES (2) (SIGMA K (: (0) N (EXP (3) K))))
I:7 (SIGMA K (: (0) N (EXP (3) K)))
O:7 (SIGMA K (: (0) N (EXP (3) K)))
O:6 (TIMES (2) (SIGMA K (: (0) N (EXP (3) K))))
O:5 (PLUS (TIMES (-3) (EXP (3) N)) (TIMES (2) (SIGMA K (: (0) N (EXP (3) K)))))
O:4 (EQUAL (PLUS (TIMES (-3) (EXP (3) N)) (TIMES (2) (SIGMA K (: (0) N (EXP (3)
K))))) (-1))
O:3 (NOT (EQUAL (PLUS (TIMES (-3) (EXP (3) N)) (TIMES (2) (SIGMA K (: (0) N (EXP
 (3) K))))) (-1)))
```

```
I:3 (NOT (EQUAL (PLUS (TIMES (-3) (EXP (3) N)) (TIMES (2) (SIGMA K (: (0) N (EXP
 (3) K))))) (-1)))
I:4 (EQUAL (PLUS (TIMES (-3) (EXP (3) N)) (TIMES (2) (SIGMA K (: (0) N (EXP (3)
K))))) (-1))
I:5 (PLUS (TIMES (-3) (EXP (3) N)) (TIMES (2) (SIGMA K (: (0) N (EXP (3) K)))))
I:6 (TIMES (-3) (EXP (3) N))
I:7 (EXP (3) N)
O:7 (EXP (3) N)
O:6 (TIMES (-3) (EXP (3) N))
I:6 (TIMES (2) (SIGMA K (: (0) N (EXP (3) K))))
I:7 (SIGMA K (: (0) N (EXP (3) K)))
O:7 (SIGMA K (: (0) N (EXP (3) K)))
O:6 (TIMES (2) (SIGMA K (: (0) N (EXP (3) K))))
O:5 (PLUS (TIMES (-3) (EXP (3) N)) (TIMES (2) (SIGMA K (: (0) N (EXP (3) K)))))
O:4 (EQUAL (PLUS (TIMES (-3) (EXP (3) N)) (TIMES (2) (SIGMA K (: (0) N (EXP (3)
K))))) (-1))
O:3 (NOT (EQUAL (PLUS (TIMES (-3) (EXP (3) N)) (TIMES (2) (SIGMA K (: (0) N (EXP
(3) K))))) (-1)))
I:3 (EQUAL (SIGMA K (: (0) (PLUS N (1)) (EXP (3) K))) (PLUS (TIMES (3) (TIMES (E
XP (2) (-1)) (EXP (3) (PLUS N (1^Z
)))))) (TIMES (-1) (EXP (2) (-1)))))
I:4 (SIGMA K (: (0) (PLUS N (1)) (EXP (3) K)))
I:5 (SUB (PLUS N (1)) (1))
O:5 (PLUS (PLUS N (1)) (MINUS (1)))
I:5 (PLUS (PLUS N (1)) (MINUS (1)))
O:5 (PLUS N (PLUS (1) (MINUS (1))))
I:5 (PLUS N (PLUS (1) (MINUS (1))))
I:6 (PLUS (1) (MINUS (1)))
I:7 (MINUS (1))
O:7 (-1)
O:6 (PLUS (0) (0))
I:6 (PLUS (0) (0))
O:6 (0)
O:5 N
O:4 (PLUS (SIGMA K (: (0) N (EXP (3) K))) (EXP (3) (PLUS N (1))))
I:4 (PLUS (SIGMA K (: (0) N (EXP (3) K))) (EXP (3) (PLUS N (1))))
I:5 (SIGMA K (: (0) N (EXP (3) K)))
O:5 (SIGMA K (: (0) N (EXP (3) K)))
I:5 (EXP (3) (PLUS N (1)))
I:6 (PLUS N (1))
O:6 (PLUS N (1))
O:5 (TIMES (EXP (3) N) (EXP (3) (1)))
I:5 (TIMES (EXP (3) N) (EXP (3) (1)))
I:6 (EXP (3) N)
O:6 (EXP (3) N)
I:6 (EXP (3) (1))
O:6 (3)
O:5 (TIMES (3) (EXP (3) N))
I:5 (TIMES (3) (EXP (3) N))
I:6 (EXP (3) N)
O:6 (EXP (3) N)
O:5 (TIMES (3) (EXP (3) N))
O:4 (PLUS (SIGMA K (: (0) N (EXP (3) K))) (TIMES (3) (EXP (3) N)))
I:4 (PLUS (SIGMA K (: (0) N (EXP (3) K))) (TIMES (3) (EXP (3) N)))
I:5 (SIGMA K (: (0) N (EXP (3) K)))
O:5 (SIGMA K (: (0) N (EXP (3) K)))
I:5 (TIMES (3) (EXP (3) N))
I:6 (EXP (3) N)
O:6 (EXP (3) N)
O:5 (TIMES (3) (EXP (3) N))
O:4 (PLUS (SIGMA K (: (0) N (EXP (3) K))) (TIMES (3) (EXP (3) N)))
I:4 (PLUS (TIMES (3) (TIMES (EXP (2) (-1)) (EXP (3) (PLUS N (1))))) (TIMES (-1)
```

```
(EXP (2) (-1))))
I:5 (TIMES (3) (TIMES (EXP (2) (-1)) (EXP (3) (PLUS N (1)))))
I:6 (TIMES (EXP (2) (-1)) (EXP (3) (PLUS N (1))))
I:7 (EXP (2) (-1))
O:7 (EXP (2) (-1))
I:7 (EXP (3) (PLUS N (1)))
I:10 (PLUS N (1))
O:10 (PLUS N (1))
O:7 (TIMES (EXP (3) N) (EXP (3) (1)))
I:7 (TIMES (EXP (3) N) (EXP (3) (1)))
I:10 (EXP (3) N)
O:10 (EXP (3) N)
I:10 (EXP (3) (1))
O:10 (3)
O:7 (TIMES (3) (EXP (3) N))
I:7 (TIMES (3) (EXP (3) N))
I:10 (EXP (3) N)
O:10 (EXP (3) N)
O:7 (TIMES (3) (EXP (3) N))
O:6 (TIMES (3) (TIMES (EXP (2) (-1)) (EXP (3) N)))
I:6 (TIMES (3) (TIMES (EXP (2) (-1)) (EXP (3) N)))
I:7 (TIMES (EXP (2) (-1)) (EXP (3) N))
I:10 (EXP (2) (-1))
O:10 (EXP (2) (-1))
I:10 (EXP (3) N)
O:10 (EXP (3) N)
O:7 (TIMES (EXP (2) (-1)) (EXP (3) N))
O:6 (TIMES (3) (TIMES (EXP (2) (-1)) (EXP (3) N)))
O:5 (TIMES (11) (TIMES (EXP (2) (-1)) (EXP (3) N)))
I:5 (TIMES (11) (TIMES (EXP (2) (-1)) (EXP (3) N)))
I:6 (TIMES (EXP (2) (-1)) (EXP (3) N))
I:7 (EXP (2) (-1))
O:7 (EXP (2) (-1))
I:7 (EXP (3) N)
O:7 (EXP (3) N)
O:6 (TIMES (EXP (2) (-1)) (EXP (3) N))
O:5 (TIMES (11) (TIMES (EXP (2) (-1)) (EXP (3) N)))
I:5 (TIMES (-1) (EXP (2) (-1)))
I:6 (EXP (2) (-1))
O:6 (EXP (2) (-1))
O:5 (TIMES (-1) (EXP (2) (-1)))
O:4 (PLUS (TIMES (11) (TIMES (EXP (2) (-1)) (EXP (3) N))) (TIMES (-1) (EXP (2) (
-1))))
I:4 (PLUS (TIMES (11) (TIMES (EXP (2) (-1)) (EXP (3) N))) (TIMES (-1) (EXP (2) (
-1))))
I:5 (TIMES (11) (TIMES (EXP (2) (-1)) (EXP (3) N)))
I:6 (TIMES (EXP (2) (-1)) (EXP (3) N))
I:7 (EXP (2) (-1))
O:7 (EXP (2) (-1))
I:7 (EXP (3) N)
O:7 (EXP (3) N)
O:6 (TIMES (EXP (2) (-1)) (EXP (3) N))
O:5 (TIMES (11) (TIMES (EXP (2) (-1)) (EXP (3) N)))
I:5 (TIMES (-1) (EXP (2) (-1)))
I:6 (EXP (2) (-1))
O:6 (EXP (2) (-1))
O:5 (TIMES (-1) (EXP (2) (-1)))
O:4 (PLUS (TIMES (11) (TIMES (EXP (2) (-1)) (EXP (3) N))) (TIMES (-1) (EXP (2) (
-1))))
O:3 (EQUAL (PLUS (MINUS (TIMES (11) (TIMES (EXP (2) (-1)) (EXP (3) N)))) (PLUS (
SIGMA K (: (0) N (EXP (3) K))) (^Z
TIMES (3) (EXP (3) N)))) (TIMES (-1) (EXP (2) (-1)))))
```

```
I:3 (EQUAL (PLUS (MINUS (TIMES (11) (TIMES (EXP (2) (-1)) (EXP (3) N)))) (PLUS (
SIGMA K (: (0) N (EXP (3) K))) (^Z
TIMES (3) (EXP (3) N)))) (TIMES (-1) (EXP (2) (-1))))
I:4 (PLUS (MINUS (TIMES (11) (TIMES (EXP (2) (-1)) (EXP (3) N)))) (PLUS (SIGMA K
 (: (0) N (EXP (3) K))) (TIMES (^Z
3) (EXP (3) N))))
I:5 (MINUS (TIMES (11) (TIMES (EXP (2) (-1)) (EXP (3) N))))
O:5 (TIMES (-11) (TIMES (EXP (2) (-1)) (EXP (3) N)))
I:5 (TIMES (-11) (TIMES (EXP (2) (-1)) (EXP (3) N)))
I:6 (TIMES (EXP (2) (-1)) (EXP (3) N))
I:7 (EXP (2) (-1))
O:7 (EXP (2) (-1))
I:7 (EXP (3) N)
O:7 (EXP (3) N)
O:6 (TIMES (EXP (2) (-1)) (EXP (3) N))
O:5 (TIMES (-11) (TIMES (EXP (2) (-1)) (EXP (3) N)))
I:5 (PLUS (SIGMA K (: (0) N (EXP (3) K))) (TIMES (3) (EXP (3) N)))
I:6 (SIGMA K (: (0) N (EXP (3) K)))
O:6 (SIGMA K (: (0) N (EXP (3) K)))
I:6 (TIMES (3) (EXP (3) N))
I:7 (EXP (3) N)
O:7 (EXP (3) N)
O:6 (TIMES (3) (EXP (3) N))
O:5 (PLUS (SIGMA K (: (0) N (EXP (3) K))) (TIMES (3) (EXP (3) N)))
O:4 (PLUS (TIMES (-3) (TIMES (EXP (2) (-1)) (EXP (3) N))) (PLUS (SIGMA K (: (0)
N (EXP (3) K))) (0)))
I:4 (PLUS (TIMES (-3) (TIMES (EXP (2) (-1)) (EXP (3) N))) (PLUS (SIGMA K (: (0)
N (EXP (3) K))) (0)))
I:5 (TIMES (-3) (TIMES (EXP (2) (-1)) (EXP (3) N)))
I:6 (TIMES (EXP (2) (-1)) (EXP (3) N))
I:7 (EXP (2) (-1))
O:7 (EXP (2) (-1))
I:7 (EXP (3) N)
O:7 (EXP (3) N)
O:6 (TIMES (EXP (2) (-1)) (EXP (3) N))
O:5 (TIMES (-3) (TIMES (EXP (2) (-1)) (EXP (3) N)))
I:5 (PLUS (SIGMA K (: (0) N (EXP (3) K))) (0))
O:5 (SIGMA K (: (0) N (EXP (3) K)))
I:5 (SIGMA K (: (0) N (EXP (3) K)))
O:5 (SIGMA K (: (0) N (EXP (3) K)))
O:4 (PLUS (TIMES (-3) (TIMES (EXP (2) (-1)) (EXP (3) N))) (SIGMA K (: (0) N (EXP
 (3) K))))
I:4 (PLUS (TIMES (-3) (TIMES (EXP (2) (-1)) (EXP (3) N))) (SIGMA K (: (0) N (EXP
 (3) K))))
I:5 (TIMES (-3) (TIMES (EXP (2) (-1)) (EXP (3) N)))
I:6 (TIMES (EXP (2) (-1)) (EXP (3) N))
I:7 (EXP (2) (-1))
O:7 (EXP (2) (-1))
I:7 (EXP (3) N)
O:7 (EXP (3) N)
O:6 (TIMES (EXP (2) (-1)) (EXP (3) N))
O:5 (TIMES (-3) (TIMES (EXP (2) (-1)) (EXP (3) N)))
I:5 (SIGMA K (: (0) N (EXP (3) K)))
O:5 (SIGMA K (: (0) N (EXP (3) K)))
O:4 (PLUS (TIMES (-3) (TIMES (EXP (2) (-1)) (EXP (3) N))) (SIGMA K (: (0) N (EXP
 (3) K))))
I:4 (TIMES (-1) (EXP (2) (-1)))
I:5 (EXP (2) (-1))
O:5 (EXP (2) (-1))
O:4 (TIMES (-1) (EXP (2) (-1)))
O:3 (AND (OR (EQUAL (2) (0)) (EQUAL (TIMES (EXP (2) (1)) (PLUS (TIMES (-3) (TIME
```

```
SIGMA K (: (0) N (EXP (3) K))))) (TIMES (EXP (2) (1)) (TIMES (-1) (EXP (2) (-1))
)))) (IF (EQUAL (2) (0)) (EQUAL ^Z
(PLUS (TIMES (-3) (TIMES (EXP (2) (-1)) (EXP (3) N))) (SIGMA K (: (0) N (EXP (3)
 K)))) (TIMES (-1) (EXP (2) (-1)^Z
)))))
I:3 (AND (OR (EQUAL (2) (0)) (EQUAL (TIMES (EXP (2) (1)) (PLUS (TIMES (-3) (TIME
S (EXP (2) (-1)) (EXP (3) N))) (^Z
SIGMA K (: (0) N (EXP (3) K))))) (TIMES (EXP (2) (1)) (TIMES (-1) (EXP (2) (-1))
)))) (IF (EQUAL (2) (0)) (EQUAL ^Z
(PLUS (TIMES (-3) (TIMES (EXP (2) (-1)) (EXP (3) N))) (SIGMA K (: (0) N (EXP (3)
 K)))) (TIMES (-1) (EXP (2) (-1)^Z
)))))
I:4 (OR (EQUAL (2) (0)) (EQUAL (TIMES (EXP (2) (1)) (PLUS (TIMES (-3) (TIMES (EX
P (2) (-1)) (EXP (3) N))) (SIGMA^Z
 K (: (0) N (EXP (3) K))))) (TIMES (EXP (2) (1)) (TIMES (-1) (EXP (2) (-1))))))
I:5 (EQUAL (2) (0))
O:5 (F)
O:4 (EQUAL (TIMES (EXP (2) (1)) (PLUS (TIMES (-3) (TIMES (EXP (2) (-1)) (EXP (3)
 N))) (SIGMA K (: (0) N (EXP (3)^Z
 K))))) (TIMES (EXP (2) (1)) (TIMES (-1) (EXP (2) (-1)))))
I:4 (EQUAL (TIMES (EXP (2) (1)) (PLUS (TIMES (-3) (TIMES (EXP (2) (-1)) (EXP (3)
 N))) (SIGMA K (: (0) N (EXP (3)^Z
 K))))) (TIMES (EXP (2) (1)) (TIMES (-1) (EXP (2) (-1)))))
I:5 (TIMES (EXP (2) (1)) (PLUS (TIMES (-3) (TIMES (EXP (2) (-1)) (EXP (3) N))) (
SIGMA K (: (0) N (EXP (3) K)))))
I:6 (EXP (2) (1))
O:6 (2)
I:6 (PLUS (TIMES (-3) (TIMES (EXP (2) (-1)) (EXP (3) N))) (SIGMA K (: (0) N (EXP
 (3) K))))
I:7 (TIMES (-3) (TIMES (EXP (2) (-1)) (EXP (3) N)))
I:10 (TIMES (EXP (2) (-1)) (EXP (3) N))
I:11 (EXP (2) (-1))
O:11 (EXP (2) (-1))
I:11 (EXP (3) N)
O:11 (EXP (3) N)
O:10 (TIMES (EXP (2) (-1)) (EXP (3) N))
O:7 (TIMES (-3) (TIMES (EXP (2) (-1)) (EXP (3) N)))
I:7 (SIGMA K (: (0) N (EXP (3) K)))
O:7 (SIGMA K (: (0) N (EXP (3) K)))
O:6 (PLUS (TIMES (-3) (TIMES (EXP (2) (-1)) (EXP (3) N))) (SIGMA K (: (0) N (EXP
 (3) K))))
O:5 (PLUS (TIMES (2) (TIMES (-3) (TIMES (EXP (2) (-1)) (EXP (3) N)))) (TIMES (2)
 (SIGMA K (: (0) N (EXP (3) K)))^Z
))
I:5 (PLUS (TIMES (2) (TIMES (-3) (TIMES (EXP (2) (-1)) (EXP (3) N)))) (TIMES (2)
 (SIGMA K (: (0) N (EXP (3) K)))^Z
))
I:6 (TIMES (2) (TIMES (-3) (TIMES (EXP (2) (-1)) (EXP (3) N))))
I:7 (TIMES (-3) (TIMES (EXP (2) (-1)) (EXP (3) N)))
I:10 (TIMES (EXP (2) (-1)) (EXP (3) N))
I:11 (EXP (2) (-1))
O:11 (EXP (2) (-1))
I:11 (EXP (3) N)
O:11 (EXP (3) N)
O:10 (TIMES (EXP (2) (-1)) (EXP (3) N))
O:7 (TIMES (-3) (TIMES (EXP (2) (-1)) (EXP (3) N)))
O:6 (TIMES (-6) (TIMES (EXP (2) (-1)) (EXP (3) N)))
I:6 (TIMES (-6) (TIMES (EXP (2) (-1)) (EXP (3) N)))
I:7 (TIMES (EXP (2) (-1)) (EXP (3) N))
I:10 (EXP (2) (-1))
O:10 (EXP (2) (-1))
I:10 (EXP (3) N)
```

```
O:10 (EXP (3) N)
O:7 (TIMES (EXP (2) (-1)) (EXP (3) N))
O:6 (TIMES (-3) (TIMES (EXP (1) (-1)) (EXP (3) N)))
I:6 (TIMES (-3) (TIMES (EXP (1) (-1)) (EXP (3) N)))
I:7 (TIMES (EXP (1) (-1)) (EXP (3) N))
I:10 (EXP (1) (-1))
O:10 (1)
O:7 (EXP (3) N)
I:7 (EXP (3) N)
O:7 (EXP (3) N)
O:6 (TIMES (-3) (EXP (3) N))
I:6 (TIMES (-3) (EXP (3) N))
I:7 (EXP (3) N)
O:7 (EXP (3) N)
O:6 (TIMES (-3) (EXP (3) N))
I:6 (TIMES (2) (SIGMA K (: (0) N (EXP (3) K))))
I:7 (SIGMA K (: (0) N (EXP (3) K)))
O:7 (SIGMA K (: (0) N (EXP (3) K)))
O:6 (TIMES (2) (SIGMA K (: (0) N (EXP (3) K))))
O:5 (PLUS (TIMES (-3) (EXP (3) N)) (TIMES (2) (SIGMA K (: (0) N (EXP (3) K)))))
I:5 (PLUS (TIMES (-3) (EXP (3) N)) (TIMES (2) (SIGMA K (: (0) N (EXP (3) K)))))
I:6 (TIMES (-3) (EXP (3) N))
I:7 (EXP (3) N)
O:7 (EXP (3) N)
O:6 (TIMES (-3) (EXP (3) N))
I:6 (TIMES (2) (SIGMA K (: (0) N (EXP (3) K))))
I:7 (SIGMA K (: (0) N (EXP (3) K)))
O:7 (SIGMA K (: (0) N (EXP (3) K)))
O:6 (TIMES (2) (SIGMA K (: (0) N (EXP (3) K))))
O:5 (PLUS (TIMES (-3) (EXP (3) N)) (TIMES (2) (SIGMA K (: (0) N (EXP (3) K)))))
I:5 (TIMES (EXP (2) (1)) (TIMES (-1) (EXP (2) (-1))))
I:6 (EXP (2) (1))
O:6 (2)
I:6 (TIMES (-1) (EXP (2) (-1)))
I:7 (EXP (2) (-1))
O:7 (EXP (2) (-1))
O:6 (TIMES (-1) (EXP (2) (-1)))
O:5 (TIMES (-2) (EXP (2) (-1)))
I:5 (TIMES (-2) (EXP (2) (-1)))
I:6 (EXP (2) (-1))
O:6 (EXP (2) (-1))
O:5 (TIMES (-1) (EXP (1) (-1)))
I:5 (TIMES (-1) (EXP (1) (-1)))
I:6 (EXP (1) (-1))
O:6 (1)
O:5 (-1)
O:4 (EQUAL (PLUS (TIMES (-3) (EXP (3) N)) (TIMES (2) (SIGMA K (: (0) N (EXP (3)
K)))))) (-1))
I:4 (EQUAL (PLUS (TIMES (-3) (EXP (3) N)) (TIMES (2) (SIGMA K (: (0) N (EXP (3)
K)))))) (-1))
I:5 (PLUS (TIMES (-3) (EXP (3) N)) (TIMES (2) (SIGMA K (: (0) N (EXP (3) K)))))
I:6 (TIMES (-3) (EXP (3) N))
I:7 (EXP (3) N)
O:7 (EXP (3) N)
O:6 (TIMES (-3) (EXP (3) N))
I:6 (TIMES (2) (SIGMA K (: (0) N (EXP (3) K))))
I:7 (SIGMA K (: (0) N (EXP (3) K)))
O:7 (SIGMA K (: (0) N (EXP (3) K)))
O:6 (TIMES (2) (SIGMA K (: (0) N (EXP (3) K))))
O:5 (PLUS (TIMES (-3) (EXP (3) N)) (TIMES (2) (SIGMA K (: (0) N (EXP (3) K)))))
O:4 (EQUAL (PLUS (TIMES (-3) (EXP (3) N)) (TIMES (2) (SIGMA K (: (0) N (EXP (3)
K)))))) (-1))
```

```
I:4 (IF (EQUAL (2) (0)) (EQUAL (PLUS (TIMES (-3) (EXP (3) N)) (TIMES (2) (SIGMA K (: (0) N (EXP (3) ^Z
K)))) (TIMES (-1) (EXP (2) (-1))))))
O:4 (OR (NOT (EQUAL (2) (0))) (EQUAL (PLUS (TIMES (-3) (EXP (3) N)) (TIMES (2) (SIGMA K (: (0) N (EX^Z
P (3) K)))) (TIMES (-1) (EXP (2) (-1))))))
I:4 (OR (NOT (EQUAL (2) (0))) (EQUAL (PLUS (TIMES (-3) (EXP (3) N)) (TIMES (2) (SIGMA K (: (0) N (EX^Z
P (3) K)))) (TIMES (-1) (EXP (2) (-1))))))
I:5 (NOT (EQUAL (2) (0)))
I:6 (EQUAL (2) (0))
O:6 (F)
O:5 (T)
O:4 (T)
O:3 (EQUAL (PLUS (TIMES (-3) (EXP (3) N)) (TIMES (2) (SIGMA K (: (0) N (EXP (3)
K))))) (-1))
I:3 (EQUAL (PLUS (TIMES (-3) (EXP (3) N)) (TIMES (2) (SIGMA K (: (0) N (EXP (3)
K))))) (-1))
I:4 (PLUS (TIMES (-3) (EXP (3) N)) (TIMES (2) (SIGMA K (: (0) N (EXP (3) K))))
I:5 (TIMES (-3) (EXP (3) N))
I:6 (EXP (3) N)
O:6 (EXP (3) N)
O:5 (TIMES (-3) (EXP (3) N))
I:5 (TIMES (2) (SIGMA K (: (0) N (EXP (3) K))))
I:6 (SIGMA K (: (0) N (EXP (3) K)))
O:6 (SIGMA K (: (0) N (EXP (3) K)))
O:5 (TIMES (2) (SIGMA K (: (0) N (EXP (3) K))))
O:4 (PLUS (TIMES (-3) (EXP (3) N)) (TIMES (2) (SIGMA K (: (0) N (EXP (3) K)))))
O:3 (EQUAL (PLUS (TIMES (-3) (EXP (3) N)) (TIMES (2) (SIGMA K (: (0) N (EXP (3)
K))))) (-1))
O:2 (T)
O:1 (T)
.
```

time = 3 seconds

Garbage collections = 0

Core Memory = 35K

Machine = Compiled Stanford LISPon a Dec10

3.2 The Proof of RI5

In this section we sketch the proof of the theorem RI5 obtained by our theorem prover. This sketch is essentially faithful to the manner in which the theorem prover proved this theorem. It differs in that many different inference steps are combined into one so as to compact the proof to a reasonable length for publication, and in that the inference steps are in a minor way permuted so as to structure the proof in a clearer manner.

It should be realized that the actual proofs obtained by this theorem prover are extremely long. For example, the actual proof of RI5 by our automatic theorem prover produces a trace on standard line printer paper over two inches thick!

$$\exists x \exists y \exists z \exists u \exists v \forall w \vdash_R \forall n \sum_{k=1}^{n} k^4 = x \cdot n^5 + y \cdot n^4 + z \cdot n^3 + u \cdot n^2 + v \cdot n + w$$

induction

$$\exists x \exists y \exists z \exists u \exists v \forall w \vdash_R \sum_{k=1}^{1} k^4 = x \cdot 1^5 + y \cdot 1^4 + z \cdot 1^3 + u \cdot 1^2 + v \cdot 1 + w$$

$$\wedge \ \forall n \left[\sum_{k=1}^{n} k^4 = x \cdot n^5 + y \cdot n^4 + z \cdot n^3 + u \cdot n^2 + v \cdot n + w \right.$$

$$\left. \sum_{k=1}^{n+1} k^4 = x \cdot (n+1)^5 + y \cdot (n+1)^4 + 2 \cdot (n+1)^3 + n \cdot (n+1)^2 + v \cdot (n+1) + w \right]$$

recursion

$$\exists x \exists y \exists z \exists u \exists v \forall w \vdash_R \{1^4 = x + y + z + u + v + w$$

$$\wedge \ \forall n \left[\sum_{k=1}^{n} k^4 = x \cdot n^5 + y \cdot n^4 + z \cdot n^3 + u \cdot n^2 + v \cdot n + w \right.$$

$$\rightarrow \sum_{k=1}^{n} k^4 + (n+1)^4 = x\cdot(n+1)^5 + y\cdot(n+1)^4 + z\cdot(n+1)^3 + u\cdot(n+1)^2 + v\cdot(n+1) + w)\}$$

fertilize

$$\exists x \exists y \exists z \exists u \exists v \exists w \vdash_R \{1 = x+y+z+u+v+w \ \wedge \ \forall n$$

$$x\cdot n^5 + y\cdot n^4 + z\cdot n^3 + u\cdot n^2 + v\cdot n + w + (n+1)^4$$

$$= x\cdot(n+1)^5 + y\cdot(n+1)^4 + z\cdot(n+1)^3 + u\cdot(n+1)^2 + v\cdot(n+1) + w\}$$

recursion

$$\exists x \exists y \exists z \exists u \exists v \exists w \vdash_R 1 = x+y+z+u+v+w \ \wedge \ \forall n$$

$$x\cdot n^5 + y\cdot n^4 + z\cdot n^3 + u\cdot n^2 + v\cdot n + 2$$

$$+ n^4 + 4n^3 + 6n^2 + 4n + 1$$

$$= x\cdot n^5 + 5xn^4 + 10xn^3 + 10xn^2 + 5xn + x$$

$$+ yn^4 + 4yn^3 + 6yn^2 + 4yn + y$$

$$+ zn^3 + 3zn^2 + 3zn + z$$

$$+ un^2 + 2un + u$$

$$+ vn + v$$

$$+ w$$

cancellation

$$\exists x \exists y \exists z \exists u \exists v \exists w \vdash_R 1 = x+y+z+u+v+w \ \wedge \ \forall n$$

$$(n^4 + 4n^3 + 6n^2 + 4n + 1) = (5xn^4 + 10xn^3 + 10xn^2 + 5xn + x)$$

$$+ (4yn^3 + 6yn^2 + 4yn + y) + (3zn^2 + 3zn + z) + (2un + u) + v$$

prepare for elimination

$$\exists x \exists y \exists z \exists u \exists v \exists w \vdash_R 1 = x+y+z+u+v+w \ \wedge \ \forall n \ (5x-1)\cdot n^4 + (10x+4y-4)\cdot n^3$$

$$+ (10x+6y+3z-6)\cdot n^2 + (5x+4y+3z+2u-4)\cdot n + (x+y+z+u+v-1)$$

elimination

$$\exists x \exists y \exists z \exists u \exists v \exists w \vdash_R 1 = x+y+z+u+v+w \ \wedge \ 5x-1=0 \ \wedge \ 10x+4y-4 = 0$$

$$\wedge \ 10x+6y+3z-6 = 0 \ \wedge \ 5x+4y+3z+2u-4 = 0 \ \wedge \ x+y+z+u+v-1 = 0$$

$$\exists x \exists y \exists z \exists u \exists v \exists w \vdash_R 1 = x+y+z+u+v+w \ \wedge \ x = 1/5 \ \wedge \ y = \frac{4-10x}{4} \ \wedge$$

$$z = \frac{6-10x-6y}{3} \ \wedge \ u = \frac{4-5x-4y-3z}{2} \ \wedge \ v = 1-x-y-z-u$$

equality subst: x

$\exists x \exists y \exists z \exists u \exists v \exists w \vdash_R w = 1-1/5-y-z-u-v \wedge x = 1/5 \wedge y = \frac{2-5(1/5)}{2}$

$\wedge z = \frac{6-10(1/5)-6y}{3} \wedge u = \frac{4-5(1/5)-4y-3z}{2} \wedge v = 1-1/5-y-z-u$

$\exists x \exists y \exists z \exists u \exists v \exists w \vdash_R w = 4/5-y-z-u-v \wedge x = 1/5 \wedge y = 1/2 \wedge z = \frac{4-6y}{3} \wedge$

$u = \frac{3-4y-3z}{2} \wedge v = 4/5-y-z-u$

equality subst: y

$\exists x \exists y \exists z \exists u \exists v \exists w \vdash_R w = 4/5-1/2-z-u-v \wedge x = 1/5 \wedge y = 1/2 \wedge z = \frac{4-6(1/2)}{3} \wedge$

$u = \frac{3-4\ 1/2-3z}{2} \wedge v=4/5-1/2-z-u$

$\exists x \exists y \exists z \exists u \exists v \exists w \vdash_R w=3/10-z-u-v \wedge x=1/5 \wedge y=1/2 \wedge z=1/3 \wedge$

$u=\frac{1-3z}{2} \wedge v=3/10-z-u$

equality subst

$\exists x \exists y \exists z \exists u \exists v \exists w \vdash_R w=3/10-1/3-u-v \wedge x=1/5 \wedge y=1/2 \wedge z=1/3 \wedge u=0 \wedge v=3/10-z-u$

$\exists x \exists y \exists z \exists u \exists v \exists w \vdash_R w-1/30-v \wedge x=1/5 \wedge y=1/2 \wedge z=1/3 \wedge u=0 \wedge v=-1/30$

$\exists x \exists y \exists z \exists u \exists v \exists w \vdash_R w=0 \wedge x=1/5 \wedge y=1/2 \wedge z=1/3 \wedge u=0 \wedge v-1/30$

metalogic laws

$\exists x \vdash_R x=1/5 \wedge \exists y \vdash_R y=1/2 \wedge \exists z \vdash_R z=1/3 \wedge \exists u \vdash_R u=0 \wedge \exists v \vdash_R v=-1/30 \wedge \exists w \vdash_R w=0$

time = $144,355_8$ms = 51.437_{10}ms = 51 sec.

Garbage collections = 10

machine = compiled stanford LISP on a DEC10.

Core Memory = 35K words

4. Conclusion

We have described a sophisticated automatic theorem prover for real algebra which is based on Symmetric Logic. We have exemplified the power of Symmetric Logic by using it to prove some long non-trivial theorems of real algebra. This provides some evidence that Symmetric Logic is of some utility in handling equation based theories.

REFERENCES

1. Robinson, J. A., "A machine-oriented logic based on the resolution principle, JACM Vol. 12 1965.

2. Bledsoe, W. W., "Splitting and reduction neuristics in automatic theorem proving, Artificial Intelligence 2, 1971.

3. Bledsoe, W. W., Boyer, K. S., and Henneman, W. H., "Computer proofs of limit theorems, Artificial Intelligence 3, 1972.

4. Pastre, D. Demonstration Antomatique de Theorems en Theorie des Ensembles, These de zeme cycle Paris VI, 1976.

5. Pastre, D., "Antomatic theorem proving in set theory," University of Paris VI, 1977.

6. Brown, F. M., "A Deductive System for Elementary Arithmetic," 2nd AISB Conference Proceedings, Edinburgh, July 1976.

7. Brown, F. M., "Doing Arithmetic without Diagrams," DAI Research Artificial Intelligence Vol. 8, 1977.

8. Brown, F. M., "A Theorem Prover for Elementary Set Theory," IJCAI5 Conference Proceedings, MIT, August 1977. Also the abstract is in the Workshop on Automatic Deduction Collected Abstracts, MIT, August 1977.

9. Brown, F. M., "Towards the Automation of Set Theory and its Logic," Artificial Intelligence, Vol. 10, 1978.

10. Brown, F. M., "Towards the Automation of Mathematical Reasoning, Thesis for Ph. D., University of Edinburgh 1977.

11. Brown, F. M., "An Automatic proof of the Completeness of Quantificational Logic," DAI Research Report 52, 1978.

12. Brown, F. M., "A Theorem Prover for Metatheory," Department of Computer Sciences, U. T. Austin, Technical Report #85 to appear in 4th Workshop on Automatic Theorem Proving Conf. Proc., Austin, Texas, 1979.

13. Brown, F. M., "A Sequent Calculus for Modal Quantificational Logic," 3rd AISB/GI Conference Proceedings, Hamburg, July 1978.

14. Bibel, W. and Schreiber J., Proof search in a Gentzers-like system of first order logic, International Computing Symposium, 1975 (North Holland Publishing Company, Amsterdam).

15. Boyer, R. S., Moore, J S., "Proving Theorems about LISP functions, IJCAI3 Proceedings, 1973.

16. Brown, F. M., "Semantical Systems for Intensional Logics based on the Modal Logic 95 + Leib. University of Texas, Department of Computer Sciences, TR 97.