Completeness Results for Inequality Provers

W. W. Bledsoe, K. Kunen, and R. Shostak

1983 ATP-65

Completeness Results for Inequality Provers

W. W. Bledsoe[1], K. Kunen[2], and R. Shostak[3]

1. Introduction

1.1 Intent

In a purely syntactical approach to automatic theorem proving, we fix some formal deductive system and input a formal theorem of that system; the computer then searches for a formal proof of that theorem. In order to make such a procedure practical, it is necessary to define the system so that there are few legal options at any given stage in a formal deduction.

For example, in standard Hilbert-style proof theory, it is conventional to allow any propositional tautology as a legal step in a formal deduction; this makes the completeness theorem fairly easy to prove. However, a computer searching mechanically for a formal proof of a theorem in such a system would waste much (or an infinite amount of) time searching through all tautologies as possible first lines of proof. Restricting the system somehow to allow only tautologies "syntactically relevant" to the theorem will shorten the search but may make the completeness theorem for the system more difficult to prove (or even false).

The system (due to Bledsoe and Hines; see [BH]) described in this paper is tailored to discover proofs of propositions about dense total orders without endpoints. In particular, it can be used to prove theorems in elementary calculus which involve the ordering of the real numbers. It is in fact an Herbrand-style rather than a Hilbert-style system, and is a modification of the resolution

1. Work supported by NSF Grant MCS 8011417.
2. Work supported by NSF Grant MCS 8200729.
3. Work supported by NSF Grant MCS 7904081.

system of J. A. Robinson (see [L] or [R]). Whereas Robinson's system was designed work in pure predicate calculus, ours involves an order (corresponding to the ordering of the real numbers), so the corresponding proof theory and completeness results are more difficult.

1.2 Lexical Issues

When describing formal languages we shall use α, β, γ, δ, ϵ, for terms, x, y, z, t for variables, f, g, h for function symbols, c, d, e, for constant symbols, $\rightarrow$, &, $\vee$, $\neg$, $\forall$, and $\exists$ for the logical operators, L for atomic formulas or literals, and A$\cdots$H for formulas or clauses. The symbol $\backslash$ signifies the end of proof, or the end of a lemma whose proof is omitted.

1.3 Orders

A PARTIAL ORDER is a relation that is transitive and irreflexive; a TOTAL ORDER is a partial order that also satisfies trichotomy. A partial order is DENSE iff it satisfies

$$\forall x \forall y(x < y \rightarrow \exists z(x < z \ \& \ z < y)).$$

A partial order is WITHOUT ENDPOINTS iff it satisfies

$$\forall x \exists y(x < y) \ \& \ \forall x \exists y(y < x).$$

Thus, the usual order of the real numbers is a dense total order without endpoints. Basic facts about orders can be found in standard texts on discrete mathematics. We shall occasionally refer to the following two facts which are somewhat less elementary. The proofs are easy exercises in constructing orders by transfinite recursion.

LEMMA 1: Suppose < partially orders a set $\mathcal{A}$. Then there is a set $\mathcal{A}'$ partially ordered by a relation <' such that $\mathcal{A}$ is a subset of $\mathcal{A}'$, < is the restriction of <' to $\mathcal{A}$, and <' is dense and without endpoints. Furthermore, if < is a total order, then <' can be taken to be total also.\

LEMMA 2: Suppose < partially orders a set $\mathcal{A}$. Then there is a total order, <' , of the SAME set $\mathcal{A}$, such that <' extends < (i.e., x < y implies x <' y). If < is dense, or without endpoints, or both, then <' can be taken to have the same properties. \

In the next section we shall discuss the model-theoretic importance of these results.

1.4 Our Predicate Logic

We assume the reader is familiar with elementary model theory; see, e.g., Enderton [E]. Let $\mathcal{L}$ be a language in predicate calculus with = whose non-logical symbols consist of the binary relation symbols < and $\leq$ (but no other relation symbols), finitely many (or 0) constant symbols, and finitely many (or 0) function symbols (of one or more places). As usual in predicate calculus, we have basic syntactic notions such as the formulas and sentences of $\mathcal{L}$, as well as semantic notions such as the notion of a structure for $\mathcal{L}$ (any non-empty set together with appropriate interpretations for the various non-logical symbols). As usual, = is interpreted as real equality in any structure. < may be interpreted as any binary relation, but we adopt the convention that $x \leq y$ MUST be interpreted as $x < y \vee x = y$. It is more common in model theory to consider $x \leq y$ merely an abbreviation of $(x < y \vee x = y)$, but in our system it will be convenient later to eliminate = in the context of orders by replacing $x = y$ by $(x \leq y \ \& \ y \leq x)$.

Recall that a formula of $\mathcal{L}$ is called UNIVERSAL iff it is of the form

$$\forall x^1 \forall x^2 \ldots \forall x^n \, A,$$

where A is quantifier-free, and a formula is called POSITIVE iff the only propositional connectives appearing in it are & and $\vee$. The two lemmas in the previous section have the following consequences.

LEMMA 1: Let A be a universal sentence of $\mathcal{L}$ that contains no function symbols. If A is valid in all dense partial orders without endpoints, then A is valid in all partial orders. Likewise for total orders.

LEMMA 2: Let A be any positive sentence of L. If A is valid in some partially ordered structure, then A is valid in some totally ordered structure. If A is valid in some dense partial order without endpoints, then A is valid in some dense total order without endpoints.

In this paper, we are primarily interested in studying those sentences which are valid in all dense total orders without endpoints; let us call such sentences DENSELY VALID. If $\mathcal{L}$ has no function symbols, then the usual decision procedure for dense total orders, (see [V]), provides a simple decision procedure for testing whether a sentence is densely valid. However, if $\mathcal{L}$ has at least one function symbol, then the set of densely valid sentences is not recursive. Of course, this set is r.e. (recursively enumerable), as can be seen from any form of the completeness theorem. The system described in this paper is intended to search for formal proofs of densely valid sentences.

A word of caution. In many applications, the intent is that '<' represent the usual order of the real numbers. Call a sentence REALLY valid

iff it is valid in all structures whose $<$ is isomorphic to the usual ordering of the reals. Densely valid and really valid are NOT the same (unless $\mathcal{L}$ has no function symbols). In part, this is because orders may fail to satisfy the least upper bound (lub) axiom. In our first-order logic, the lub axiom is really an infinite schema; namely, for each formula B(x) (with possibly other free variables besides x), we have the universal closure of the following sentence, which says that if $\{x: B(x)\}$ is non-empty and bounded above, its lub exists:

$$(\exists x B(x) \ \& \ \exists y \forall x(B(x) \rightarrow x \leq y)) \rightarrow$$

$$\exists y[(\forall x(B(x) \rightarrow x \leq y) \ \&$$

$$\forall z(z < y \rightarrow \exists x(B(x) \ \& \ z < x))]$$

For any B, this axiom is really valid, but for very simple B it fails to be densely valid; for example, let B(x) be $f(x) = x$; this fails in the rationals if f is any function such that $f(x) = x$ iff $x < \sqrt{2}$. Since most theorems of elementary calculus require the lub axiom, the use of this axiom must be put in explicitly (by the human) in the input. Thus, if A is the desired theorem, the input to the prover will be of the form

$$C_1 \& \ .. \ \& C_n \rightarrow A,$$

where C_1 .. C_n are the relevant instances of the lub axiom.

A more fundamental (and less interesting practically) distinction between really valid and densely valid is the following. The set of really valid sentences is not r.e. (in fact it is a complete π_1^2 set). Thus, it has no recursive axiomatization. Specifically, let LUB be the set of instances of the lub

axiom. LUB is recursive, so the set LUB* of sentences true in all dense total orders without endpoints that satisfy LUB is r.e. Thus, there are really valid sentences that are not in LUB*; a suitable encoding of the consistency of LUB is such a sentence; a suitable encoding of Borel determinacy is another. However, LUB has the same proof-theoretic strength as second-order number theory. Thus, all the theorems of calculus at the level of difficulty currently being investigated in automatic theorem proving certainly are in LUB*, and the practical interest presently is in getting the computer to prove these theorems efficiently (or at all), rather than in looking at more advanced theorems for which LUB does not suffice.

1.5 Herbrandization

The official versions of all our theorems will be stated within the Herbrand-style framework of resolution theory. We describe here how to translate into this framework from more standard predicate logic. Let A be a sentence of $\mathcal{L}$, as above. Our original desire is to design a system that will test whether A is densely valid; more precisely, it will input A and terminate in a finite amount of time iff A is densely valid, in which case the output will be a formal proof of A. Call a sentence B DENSELY INCONSISTENT if B is false in all structures whose $<$ is a dense total order without endpoints; equivalently, iff $\neg$ B is densely valid. It will be convenient to view our procedure as one that tests whether $\neg$ A is densely inconsistent. Furthermore, before beginning to work on $\neg$ A, we shall replace $\neg$ A by another sentence which is in a very simple logical form.

We first define some terminology. A LITERAL is a formula of the form $\alpha < \beta$ or $\alpha \leq \beta$, where a and b are terms. A CLAUSE is a finite set of

literals. We use $\square$ to denote the empty clause. If C is the clause

$$\{L_1, L_2, \ldots, L_m\}$$

and $\mathcal{M}$ is a structure, we say that C is valid in $\mathcal{M}$ iff the formula

$$L_1 \vee L_2 \quad \ldots \vee L_m$$

is valid in $\mathcal{M}$ in the usual sense in predicate calculus (i.e., its universal quantification is true). It is consistent with our notation to say (by definition) that the empty clause, $\square$, is never valid in $\mathcal{M}$.

Since our semantics causes a clause to be interpreted as the disjunction of its literals, we shall often use the notation

$$L_1 \vee L_2 \vee \ldots \vee L_m$$

as an abbreviation for the clause

$$\{L_1, L_2, \ldots, L_m\}$$

and we shall use $C \vee D$ to abbreviate $C \cup D$. But note that

$$L_1 \vee L_2,$$

$$L_2 \vee L_1, \text{ and}$$

$$L_2 \vee L_1 \vee L_2$$

are all the SAME clause, namely

$$\{L_1, L_2\}.$$

If $\mathcal{A}$ is a set of clauses, we say that $\mathcal{A}$ is valid in $\mathcal{M}$ iff every member of $\mathcal{A}$ is valid is $\mathcal{M}$. $\mathcal{A}$ is DENSELY INCONSISTENT iff there is no $\mathcal{M}$ such that $\mathcal{A}$ is valid in $\mathcal{M}$ and $\mathcal{M}$ is a dense total ordering without endpoints. Note that besides being in simple logical form, $\mathcal{A}$ does not use negation or equality. Our proof theory will be a collection of rules deriving semantic consequences of $\mathcal{A}$; the completeness theorem will say that if $\mathcal{A}$ is densely inconsistent, the proof rules will eventually derive $\square$ from $\mathcal{A}$.

We now describe the procedure by which, given a sentence A, we find a finite set $\mathcal{A}$ of clauses such that A is densely valid iff $\mathcal{A}$ is densely inconsistent. The procedure is essentially one of Skolemization and reduction to conjunctive normal form. In particular, $\mathcal{A}$ may involve new function symbols not used in A.

Let B be the sentence of $\mathcal{L}$ obtained by replacing all subformulas of A of the form $\alpha = \beta$ (where α and β are terms) by $(\alpha \leq \beta \ \& \ \beta \leq \alpha)$. Then B does not use =, and B is equivalent to A in all structures in which $<$ is irreflexive.

Let C be $\neg$B. Clearly, B is densely valid iff C is densely inconsistent. By Skolemization, we can find a language $\mathcal{L}^+$ formed by adjoining finitely many function and constant symbols to $\mathcal{L}$ and a sentence D of $\mathcal{L}^+$ such that D is universal and such that D is densely inconsistent iff C is.

We continue to operate on D. Our theory does not distinguish between the "interesting" functions symbols of $\mathcal{L}$ and the "artificial" Skolem functions adjoined in $\mathcal{L}^+$. Say D is

$$\forall x_1 \forall x_2 \ldots x_n \, D^{\square},$$

where $D^{\square}$ is quantifier-free. Let $E^{\square}$ be obtained by reducing $D^{\square}$ to conjunctive normal form; so, $E^{\square}$ is propositionally equivalent to $D^{\square}$, and $E^{\square}$ is a conjunction of disjunctions of atomic and negated atomic formulas. We have already eliminated $=$, so atomic formulas are all of the form $\alpha < \beta$ and $\alpha \leq \beta$. Let $F^{\square}$ be obtained from $E^{\square}$ by replacing formulas of the form $\neg(\alpha < \beta)$ by $\beta \leq \alpha$ and replacing $\neg(\alpha \leq \beta)$ by $\beta < \alpha$. Then $F^{\square}$ is a conjunction of disjunctions of literals. Say $F^{\square}$ is

$$F_1 \ \& \ F_2 \ \& \ \ldots \ \& \ F_m,$$

where each F_i is

$$L_{i1} \vee \ldots \vee L_{in_i} .$$

Let C_i be the clause

$$\{L_{i1}, \ldots, L_{in_i}\} ,$$

and let $\mathcal{A}$ be

$$\{C_1, C_2, \ldots, C_m\}$$

Then A is densely valid iff $\mathcal{A}$ is densely inconsistent.

2. Some Rules of Inference

2.1 Intent

In this section we describe some rules of inference. Each rule will obviously be sound; i.e., for any given structure in which $<$ is interpreted as a dense total order without endpoints, it will lead from clauses valid in that structure to a clause valid in that structure. It will also be a fairly trivial modification of Herbrand's Theorem to see that all our rules together will lead to a complete system. However, our final deductive system will not allow all the rules, but will be restricted so that only a few of them can apply at any particular time.

Of course, in theory we do not need a new system at all, since Robinson's original resolution system works for any language in predicate logic. Thus, we could write a set of clauses, $\mathcal{DO}$, expressing the axioms for dense total orders without endpoints; then, if a set $\mathcal{A}$ of clauses is densely inconsistent, ordinary resolution will produce $\square$ from $\mathcal{A}$ union $\mathcal{DO}$. However, since $\mathcal{DO}$ is a rather long list, this approach would not really be practical. Rather, we shall modify resolution to incorporate the properties of such orders as proof rules rather than axioms, so that the only axioms which will need to be explicitly added to $\mathcal{A}$ will be the axioms asserting that equals may be substituted for equals within a function symbol.

2.2 Substitution and Renaming

We shall define a SUBSTITUTION FUNCTION to be any function whose domain is the set of all variable symbols and whose range is a subset of the set of terms of $\mathcal{L}$. If σ is a substitution function, we consider it to

operate on the terms, literals, and clauses of $\mathcal{L}$ in the obvious way. As usual in predicate logic, we assume that the set of all variable symbols is countably infinite, so σ always has infinite domain; of course, in any particular application of σ, we are only interested in finitely many values.

It is clear that substitution is a valid rule of inference; i.e., if a clause C is valid in a structure and σ is any substitution function, then $C\sigma$ is also valid in that structure (we are following the usual convention here of writing substitution functions on the right). It is also clear that since there are infinitely many substitution instances of C, one should not allow arbitrary substitution as a rule of inference in our system. In fact, we shall use substitutions only when they occur as a renaming or as an mgu; we take up these two concepts next.

A RENAMING is a substitution function which maps the variables 1-1 onto the set of variables. It will be safe to allow arbitrary renamings as a rule of inference. Even though a clause C has infinitely many renamings, they are all of exactly the same form as C, so that the prover never needs to list any of them, but merely has to bear in mind that they exist.

If α_1 and α_2 are terms, they are said to be UNIFIABLE iff there is a substitution σ such that $\alpha_1\sigma = \alpha_2\sigma$, and we say that σ UNIFIES a_1 and a_2. σ is a MOST GENERAL UNIFIER (mgu) of a_1 and a_2 iff σ unifies a_1 and a_2 and for any τ which unifies a_1 and a_2, there exists a ν such that $\tau = \sigma\nu$ (the substitution σ followed by ν); informally, σ makes the minimum changes necessary to unify a_1 and a_2. It is known (see, e.g., [L] or [R]) that if a_1 and a_2 are unifiable, then they have an mgu that is unique up to renaming, in that if σ and τ are both mgu's of a_1 and a_2, then for some renaming γ, $\sigma = \tau\gamma$. One

usually adopts some convention for choosing a specific mgu and writes $\sigma = \text{mgu}(a_1, a_2)$; this will not cause trouble in our deductive system since renaming is allowed anyway as a proof rule.

One may also unify more than two terms. For example, we say that a_1, a_2, a_3 are unifiable iff there is a σ such that $a_1\sigma = a_2\sigma = a_3\sigma$. In that case there is a most general unifier; in fact if $\tau = \text{mgu}(a_2, a_3)$, and $\nu = \text{mgu}(a_1\tau, a_2\tau)$, then $\text{mgu}(a_1, a_2, a_3) = \tau\nu$.

Precisely the same discussion pertains to unifying literals instead of terms. In fact, on this purely syntactic level, there is no distinction between the function symbols occurring in terms and the predicate symbols ($<$ and $\leq$) occurring in literals.

2.3 Self-Chaining and Factoring

We describe here two instances of substitution which we shall allow because they decrease the length of a clause. Self-chaining is used when a substitution instance of C contains an inconsistent literal (e.g., $\alpha < \alpha$), which may then be deleted from the disjunction, and factoring is used when a substitution instance of C contains a redundancy, such as $\alpha < \beta \vee \alpha < \beta$, which may then be shortened to $\alpha < \beta$.

Suppose C is the clause

$$\alpha < \beta \vee D$$

where the terms α and β are unifiable. If σ is $\text{mgu}(\alpha, \beta)$, then $C\sigma$ is logically equivalent to $D\sigma$. We call $D\sigma$ a result of self-chaining applied to C.

For factoring, suppose that the literals L_1 and L_2 are unifiable, and let σ be their mgu. Let $L = L_1\sigma = L_2\sigma$. If C is the clause

$$L_1 \vee L_2 \vee D,$$

then $C\sigma$ is the clause

$$L \vee D\sigma;$$

note that by our convention, $L \vee L$ abbreviates the set $\{L,L\}$ which equals $\{L\}$. We call $C\sigma$ a result of factoring applied to C.

There is another possible form of factoring which is not necessary, so that we shall NOT include it as part of our formal system. Say C is

$$\alpha_1 < \beta_1 \vee \alpha_2 < \beta_2 \vee D.$$

Assume that the literals $\alpha_1 < \beta_1$ and $\alpha_2 < \beta_2$ are unifiable, with σ their mgu and $\alpha = \alpha_1\sigma$ and $\beta = \beta_1\sigma$. Then $C\sigma$ is

$$\alpha < \beta \vee \alpha \leq \beta \vee D\sigma$$

which is not any shorter than C, but which is equivalent in any ordered structure to

$$\alpha \leq \beta \vee D\sigma.$$

One might call this last clause a result of factoring applied to C. We shall not, since we shall obtain completeness without this rule, but as a practical matter, it is probably advisable to include this rule in actual provers. Of course adding this rule (or any other sound rule of inference) to a complete

system will maintain completeness.

We remark that multiple self-chainings and factorings are obtained by applying these rules several times. For example, suppose C is

$$L_1 \vee L_2 \vee L_3 \vee D,$$

$\sigma = \mathrm{mgu}(L_1, L_2, L_3)$, and $L_1\sigma$ is L. Then we may obtain

$$L \vee D\sigma$$

by factoring twice; specifically, let $\tau = \mathrm{mgu}(L_2, L_3)$, and let $\nu = \mathrm{mgu}(L_1\tau, L_2\tau)$. Then $\sigma = \tau\nu$, so we factor first using τ and then again using ν.

2.4 Chaining

Chaining operates on two clauses and uses transitivity of $\prec$; for example, from $\alpha < \beta$ and $\beta < \gamma$ we infer $\alpha < \gamma$. More generally, if β_1 and β_2 are unifiable and σ is their mgu, we wish to infer $(\alpha < \gamma)\sigma$ from $\alpha < \beta_1$ and $\beta_2 < \gamma$. There is a slight complication introduced here by the fact that variables occurring in different clauses could just as well be distinct. For example, the terms f(x,c) and f(d,x) are NOT unifiable, and we are certainly not justified in concluding $\square$ from the clause $f(x,c) < f(d,x)$ as we would from $f(x,c) < f(d,y)$. However, from the two clauses.

1. $e_1 < f(x,c)$
2. $f(d,x) < e_2$,

we would be justified in concluding $e_1 < e_2$, since this is valid in any ordered structure in which (1) and (2) are valid; (2) could just as well have been $f(d,y) < e_2$.

One way to handle this would be to allow different substitutions to act on the different clauses; it is certainly sound to conclude $\alpha\sigma < \gamma\tau$ from $\alpha < \beta_1$ and $\beta_2 < \gamma$ if $\beta_1\sigma$ is the same as $\beta_2\tau$. However, this is not enough. Consider

1. $f(x) \leq c$
2. $c \leq f(x)$.

Here there is no problem with unification; whether we chain on the c or the $f(x)$, the mgu is the identity substitution; but all we can produce are the trivialities $f(x) \leq f(x)$ and $c \leq c$. We should really be able to chain on the c and produce $f(x) \leq f(y)$ (which asserts that f is a constant function); e.g., the x in clause 1 should really be considered a DIFFERENT VARIABLE from the x in clause 2.

The way this is handled is to STANDARDIZE APART variables. Informally, this means that distinct clauses are considered to have disjoint sets of variables. Formally, we handle this by requiring that chaining may apply only to pairs of clauses which have no variables in common. This will not cause any undue restriction, since we are allowing arbitrary renaming as a rule of inference. As mentioned above, the computer never attempts to list all the renamings of a clause, C. However, when examining the possible chaining consequences of C and D, it will first choose a renaming, C', of C with variables disjoint from the ones in D (it does not matter which renaming is chosen), and then list all the (finite number of) chaining consequences of C' and D.

We must also be careful, in defining chaining, to allow for $\leq$ as well as $<$. Thus if $@_1$ and $@_2$ are either of the symbols $<$ and $\leq$, we wish to be able

to apply transitivity to $\alpha @_1 \beta$ and $\beta @_2 \gamma$ to arrive at a literal of the form $\alpha \ @ \ \beta$. Define $tr(@_1,@_2)$ to be the symbol $<$ iff at least one of $@_1$ and $@_2$ is $<$, and the symbol $\leq$ otherwise.

Now, let C be the clause,

$$\alpha \ @_1 \ \beta \ \vee \ C',$$

and let D be the clause,

$$\gamma \ @_2 \ \delta \ \vee \ D' \ ,$$

and let $@$ be $tr(@_1,@_2)$. Assume that β and γ are unifiable, and let σ be their mgu. Then we say the clause

$$(\alpha \ @ \ \delta \ \vee \ C' \ \vee \ D')\sigma$$

is a result of chaining C and D on the terms b and c, PROVIDED that C and D have no variables in common.

2.5 Variable Elimination

Our proof rules have not yet reflected the fact that we wish to consider only dense total orders without endpoints. Thus, for example, the clause $x \leq c$ is densely inconsistent, but is valid in totally ordered structures with a largest element (if the constant symbol c is interpreted as that element); since the proof rules discussed so far are sound for such structures, they will never derive $\square$ from this clause. Note that there is no such problem with $x < c$, which is not valid in any ordered structure, and which yields $\square$ via self-chaining.

The deduction rule called "variable elimination" really corresponds to what a model-theorist would call the QUANTIFIER elimination procedure for dense total orders without endpoints. This is the procedure by which the quantifiers are successively eliminated from a formula, eventually showing that the formula is equivalent (in all such orders) to a quantifier-free formula; one hence obtains a decision procedure for validity in such orders. Of course, this procedure works only if the language has no function symbols.

DEFINITION: A variable x is ELIGIBLE in a clause C iff x occurs in C, but no term of C other than x itself contains x.

If we phrase quantifier elimination in our framework, we have the following: Suppose the variable x is eligible in the clause C. Since x may be thought of as being universally quantified, we may attempt to eliminate it and obtain an equivalent (but simpler) clause which does not use x.

There are four types of variable elimination. The first three correspond to the three facts that our orders have no largest element, no smallest element, and are dense.

For type 1, suppose that C is of the form

$$x@_1\alpha_1 \vee x@_2\alpha_2 \ldots \vee x@_n\alpha_n \vee D,$$

where x does not occur anywhere in D or in the terms $\alpha_1, \alpha_2, \ldots, \alpha_n$. Then we say that D is the result of type 1 variable elimination applied to C. D may be empty, so that, for example, we may now infer $\square$ directly from $x < c \vee x \leq d$. The symbols $@_1, @_2, \ldots, @_n$ are free to be $<$ of $\leq$ in any combination.

Type 2 variable elimination is the exact analogue of type 1, applied to clauses of the form

$$\alpha_1@_1x \vee \alpha_2@_2x \ldots \vee \alpha_n@_nx \vee D,$$

with the same restrictions on occurrences of x.

For type 3, suppose C is of the form

$$\alpha_1 @_1 x \vee \alpha_2 @_2 x \ldots \vee \alpha_n @_n x \vee$$

$$x \$_1 \beta_1 \vee x \$_2 \beta_2 \ldots \vee x \$_m \beta_m \vee D,$$

where neither n nor m is 0, and x does not occur in D or in any of the α_i or β_j. Let $\&_{ij}$ be the symbol $<$ iff both $@_i$ and $\$_j$ are $<$; otherwise $\&_{ij}$ is $\leq$ (note that we do not use $tr(@_i,\$_j)$ here). Let L_{ij} be the literal $\alpha_i \&_{ij} \beta_j$, and let E be the disjunction of D with all of the L_{ij}; namely:

$$\begin{array}{l} L_{11} \vee L_{12} \ldots \vee L_{1n} \vee \\ \quad \ldots\ldots\ldots \vee \\ L_{n1} \vee L_{n2} \ldots \vee L_{nm} \vee D. \end{array}$$

Then we say that E is a result of type 3 variable elimination applied to C. E may well be longer than C, although it is somehow "simpler", since it contains one fewer variable.

Finally, if C is any clause of the form $x < x \vee D$, we shall say that D is a consequence of C by type 4 variable elimination. Of course, we could also apply self-chaining here, but we shall eventually restrict self-chaining (see Section 4.3) so that it does not apply.

The fact that variable elimination is a sound rule may be stated officially by the following easy lemma:

LEMMA 1: Suppose that the clause E is obtained from C by variable elimination. Then for any dense partial order without endpoints, C is valid in that order iff E is.

If x is eligible in C and C does not contain any literal of the form $x \leq x$, then one of the four types of variable elimination may be applied to eliminate x from C. An easy induction now shows:

LEMMA 2: For any clause C, variable elimination may be applied finitely many times to C to obtain a clause D such that either D has no eligible variables or D contains a literal of the form $x \leq x$.

For example, if C is $x \leq y \vee y \leq x$, then D is $x \leq x$; in fact, one can design the prover to simply throw out tautologies such as C and D, but care must be taken if the system is to remain complete; see Section 5.1. In the course of obtaining D, one may eliminate variables which were not eligible in C, but which become eligible after other variables are eliminated. For example, if C is $x < f(y) \vee y < c$, we may eliminate x first to obtain $y < c$, and then eliminate y to obtain $\square$.

2.6 Remarks on Trichotomy

The reader may note that we have not yet introduced a proof rule to reflect trichotomy, so that our proof theory might be relevant to partial orders, not total orders. In fact, however, trichotomy was incorporated by our herbrandization process. Specifically, trichotomy was used in replacing $\neg(\alpha < \beta)$ by $\beta \leq \alpha$ and $\neg(\alpha \leq \beta)$ by $\beta < \alpha$. This step is not valid for partial orders. It is now not necessary to adopt any further axioms (such as $x \leq y \vee y < x$) or proof rules for trichotomy.

More formally, that no such axioms or rules are needed is simply to say that we shall succeed in proving the completeness theorem without them.

However, this fact is not tied to some specific feature of our particular deductive system. If $\mathcal{A}$ is a set of clauses, $\mathcal{A}$ corresponds to a positive sentence, A, in ordinary predicate logic; namely, A is the universal quantification of the conjunction of the clauses in $\mathcal{A}$. By Lemma 2 of Section 1.4, A (and hence $\mathcal{A}$) fails to be valid in any dense total order without endpoints iff it fails to be valid in any dense partial order without endpoints. Thus, to derive $\square$ from $\mathcal{A}$, we would not expect that our proof theory would have to contain any rule reflecting trichotomy.

Observe that unlike Hilbert-style proof theories, our completeness theorem will involve deriving $\square$ only from inconsistent $\mathcal{A}$; it will not be the case that if C is a semantic consequence of $\mathcal{A}$, then C is derivable from $\mathcal{A}$. In particular, the trichotomy rule will not be derivable from the empty set of clauses; for that matter, neither will anything else.

2.7 Substitution of Equals for Equals

We shall need to incorporate the fact that equals may be substituted for equals in an expression. For a simple example, consider the clauses:

1. $c \leq d$
2. $d \leq c$
3. $f(c) < f(d)$

where c and d are constant symbols. One cannot derive $\square$ from these clauses using the rules presented so far; in fact, it is easy to check that the only new clauses which can be derived are $c \leq c$ and $d \leq d$. We handle this by explicitly adding an axiom. In general, if f is an n-place function symbol, we define

the clause EE(f) to be:

$$x_1 < y_1 \vee \ldots \vee x_n < y_n \vee$$

$$y_1 < x_1 \vee \ldots \vee y_n < x_n \vee$$

$$f(x_1,\ldots,x_n) \leq f(y_1,\ldots,y_n).$$

Let $\mathcal{EE}$ be the set of all EE(f) for f in our language. Then our completeness theorem will actually say that if a set S of clauses is densely inconsistent, then there is a deduction of $\square$ from S union $\mathcal{EE}$.

Now, to consider our specific example, we may add EE(f), which is

4. $x < y \vee y < x \vee f(x) \leq f(y)$.

We proceed to derive $\square$ as follows:

5. $d < y \vee y < d \vee f(c) < f(y)$ (3,4, chaining d/x)

6. $d < c \vee c < d$ (5, self-chaining c/y)

7. $c < c \vee c < d$ (1,6, chaining)

8. $c < c$ (2,7, chaining)

9. $\square$ (8, self-chaining)

As was the case with trichotomy, our system does not need any special rules or axioms reflecting the fact that equality is symmetric, reflexive, and transitive. These facts are subsumed by the rules for $\leq$.

3. The Ground Case

3.1 Basics

Informally, a set of clauses is ground consistent iff it is consistent when we regard each term as a constant symbol. Formally, let $\mathcal{A}$ be a set of clauses. A ground model for $\mathcal{A}$ is a triple $(\mathcal{Q}, <, F)$, where $\mathcal{Q}$ is a set partially ordered by the relation $<$ and F is a function from the set of terms occurring in $\mathcal{A}$ into $\mathcal{Q}$ such that all the clauses of $\mathcal{A}$ are true in the obvious sense; namely, for each C in $\mathcal{A}$, either C contains a literal of the form $\alpha < \beta$ and we have $F(\alpha) < F(\beta)$, or C contains a literal of the form $\alpha \leq \beta$ and we have $F(\alpha) \leq F(\beta)$. We say $\mathcal{A}$ is ground consistent iff $\mathcal{A}$ has a ground model; if not, $\mathcal{A}$ is ground inconsistent.

A term, literal, or clause is said to be GROUND iff it contains no variables. In the definition of ground consistent, we are treating the terms of $\mathcal{A}$ AS IF they were distinct and unrelated ground terms even if in fact they contain variables.

Clearly, if $\mathcal{A}$ has any totally ordered model, it is ground consistent. The converse of this is false; for example, let $\mathcal{A}$ be $\{x < y\}$. Even if $\mathcal{A}$ contains only ground clauses, it may be ground consistent but fail to have any totally ordered model; for example, let $\mathcal{A}$ contain the clauses $c \leq d$, $d \leq c$, and $f(c) < f(d)$.

Observe that here it does not matter whether we are looking at partial orders, total orders, or dense total orders:

LEMMA 1: Let $\mathcal{A}$ be a set of clauses. Then the following are equivalent:

1. $\mathcal{A}$ has a ground model.
2. $\mathcal{A}$ has a ground model $(\mathcal{Q},<,F)$ in which $<$ totally orders $\mathcal{Q}$.
3. $\mathcal{A}$ has a ground model $(\mathcal{Q},<,F)$ in which $<$ totally orders $\mathcal{Q}$ and $<$ is dense and without endpoints.
4. $\mathcal{A}$ has a ground model $(\mathcal{Q},<,F)$ in which $<$ totally orders $\mathcal{Q}$ and F maps the terms occurring in $\mathcal{A}$ ONTO $\mathcal{U}$.

PROOF: It is sufficient to show that (1) implies (2-4). If $(\mathcal{Q},<,F)$ is any ground model for $\mathcal{A}$, we may find a relation, $<'$ on $\mathcal{Q}$ which extends $\prec$ and which totally orders $\mathcal{Q}$ (Lemma 2 of Section 1.3). Then $(\mathcal{Q},<',F)$ satisfies (2). Now, we can construct a superset, B, of $\mathcal{Q}$, and a dense total order without endpoints, $<''$ of B such that $<''$ restricted to $\mathcal{Q}$ agrees with $<'$ (Lemma 1 of Section 1.3). Then $(B,<'',F)$ satisfies (3). Finally, let C be the range of F and let $<'''$ be the restriction of $<'$ to C. Then $(C,<'''.F)$ satisfies (4).

Of course, one cannot in general expect to get an $(\mathcal{Q},<,F)$ to satisfy (3) and (4) simultaneously; for example, if $\mathcal{A}$ is finite, then (4) would imply that $\mathcal{Q}$ is finite, so the ordering could not be dense. We also remark that the usual compactness theorem yields:

LEMMA 2: Let $\mathcal{A}$ be any set of clauses. $\mathcal{A}$ is ground consistent iff every finite subset of $\mathcal{A}$ is ground consistent.

We now take up the following two topics: First, there is the analog of Herbrand's Theorem from ordinary predicate logic, which says that if a set of clauses is inconsistent, then some substitution instance of them is ground inconsistent. Second, there is a ground completeness theorem.

3.2 Herbrand's Theorem

If $\mathcal{A}$ is any set of clauses of the language $\mathcal{L}$, let subinst($\mathcal{A}$) be the set of all $C\sigma$ such that C is a clause in $\mathcal{A}$ and σ is a substitution function of $\mathcal{L}$. The expected Herbrand theorem would be that if $\mathcal{A}$ is densely inconsistent, then subinst($\mathcal{A}$) is ground inconsistent, but this is false for two reasons. First, we must assume that $\mathcal{A}$ contains the set $\mathcal{EE}$ of Section 2.7. Second, we wanted dense orders. For example, if $\mathcal{A}$ consists only of the two clauses $c < d$ and $x \leq c \vee d \leq x$, then $\mathcal{EE}$ is vacuous and subinst($\mathcal{A}$) is ground consistent, but every model for $\mathcal{A}$ has no elements between P and Q. For an Herbrand theorem which produces a dense model for $\mathcal{A}$, we would need to assume that $\mathcal{A}$ contains the axioms for dense total order without endpoints. More on this later. For the time being, we shall only state the Herbrand result in terms of totally ordered models.

THEOREM: If subinst($\mathcal{A}$) is ground consistent and $\mathcal{EE}$ is a subset of $\mathcal{A}$, then $\mathcal{A}$ has a totally ordered model.

PROOF: Let $(\mathcal{Q},<,F)$ be a ground model for subinst($\mathcal{A}$). As we saw above, we may assume that F maps the terms of $\mathcal{L}$ onto $\mathcal{Q}$ and $<$ totally orders $\mathcal{Q}$. We shall make $(\mathcal{Q},<)$ into a model for $\mathcal{A}$. If c is a constant symbol of $\mathcal{L}$, interpret c as $F(c)$. If g is an n-place function symbol of $\mathcal{L}$, the interpretation of g in the model should be an n-place function, G, on $\mathcal{Q}$. We would like to define G by

$$G(F(\alpha_1), \ldots, F(\alpha_n)) = F(g(\alpha_1, \ldots, \alpha_n)).$$

In view of the fact that F is onto, this is a legitimate definition PROVIDED we can show that if $\beta_1,\ldots,\beta_n$ are terms with each $F(\alpha_1) = F(\beta_1)$ then

$$F(g(\alpha_1, \ldots, \alpha_n)) = f(g(\beta_1, \ldots, \beta_n)).$$

Suppose that this fails; say

$$f(g(\alpha_1, \ldots, \alpha_n)) > f(g(\beta_1, \ldots, \beta_n)).$$

Since the axiom EE(g) is in $\mathcal{A}$, its substitution instance,

$$\alpha_1 < \beta_1 \vee \ldots \vee \alpha_n < \beta_n \vee$$
$$\beta_1 < \alpha_1 \vee \ldots \vee \beta_n < \alpha_n \vee$$
$$g(a_1,\ldots,a_n) \leq g(b_1,\ldots,b_n),$$

is a subinst($\mathcal{A}$). Since the ground model must make one of the literals in this last clause true, we have

$$F(g(\alpha_1, \ldots, \alpha_n)) \leq F(g(\beta_1, \ldots, \beta_n)),$$

a contradiction.

Now, as usual in completeness theorems, we easily check, using the fact that F is onto, that we have constructed a totally ordered model for $\mathcal{A}$.

3.3 The Ground Completeness Theorem

We now show that if $\mathcal{A}$ is ground inconsistent, we can derive $\square$ from $\mathcal{A}$ by using only ground instances of chaining and self-chaining (that is,

the only substitution allowed is the identity substitution). Furthermore, we show that there is a great amount of freedom in the order in which we eliminate terms. We may fix any set $\mathcal{T}$ of terms and demand that we first do only chainings on terms in $\mathcal{T}$, until we obtain a ground inconsistent set of clauses that do not use $\mathcal{T}$ at all.

First, we state specifically our rules of inference. If C is the clause

$$\beta < \beta \vee D,$$

then we call D a result of ground self-chaining applied to C on the term b. For chaining, if C is the clause

$$\alpha @_1 \beta \vee C',$$

D is the clause,

$$\beta @_2 \delta \vee D'$$

and @ is $tr(@_1, @_2)$ (see Section 2.4), then we say that the clause

$$\alpha @ \delta \vee C' \vee D'$$

is a result of ground chaining C and D on the term b. Observe that in this definition, we do not require the variables of C and D to be disjoint. This is in line with our definition of ground consistent, in which we treat all terms as if they were ground.