CLAUSAL INTUITIONISTIC LOGIC
II. TABLEAU PROOF PROCEDURES

L. THORNE McCARTY

Clausal intuitionistic logic is an extension of Horn-clause logic which permits the appearance of negations and embedded implications on the right-hand side of a rule, and interprets these new rules intuitionistically in a set of partial models. In this article the second of a pair, clausal intuitionistic logic, is shown to have a tableau proof procedure that generalizes Horn-clause refutation proofs. The proof procedures are explained by means of several detailed examples. Soundness and completeness theorems are stated and proven in full detail.

INTRODUCTION

Since the advent of Horn-clause logic programming in the mid 1970s, there have been numerous attempts to extend the expressive power of Horn-clause logic while preserving some of its attractive computational properties. In the first paper of this pair, we introduced a language that extends Horn-clause logic by adding the following rules for negations:

\[ P(x) \leftarrow \neg Q(x; y), \]
\[ P(x) \Rightarrow \neg Q(x; y) \]

and the following rules for embedded implications:

\[ P(x) \leftarrow [Q(x; y) \Rightarrow R(x; y)], \]
\[ P(x) \Rightarrow [Q(x; y) \Rightarrow R(x; y)] \]

where the variables \( y \) are given an implicit universal quantification with scope extending just outside the right-hand side of the rule. Our proposal was to interpret these new rules intuitionistically, rather than classically, and under this interpretation we were able to develop a fixed-point semantics for negations and embedded
implications that generalizes the standard fixed-point semantics for Horn clauses [29, 1]. Specifically, letting $B = \mathcal{P}(H)$ be the powerset lattice over the Herbrand base, let $J_0$ be the largest subset of $B$ that satisfies the Horn clauses $P \Leftarrow \wedge Q_j$ and $P \Leftarrow \vee Q_j$ and the new rules in the form $P \Rightarrow \neg Q$ and $P \Rightarrow [Q \Rightarrow R]$, and letting $T_N$ be a certain monotonic transformation associated with the new rules in the form $P \Leftarrow \neg Q$ and $P \Leftarrow [Q \Rightarrow R]$, we constructed the greatest fixed point $K^*$ of the transformation $T(J) = T_N(J) \cap J_0$, and we showed (in Theorem 3.7) that $K^*$ is the unique maximal Kripke model for the total set of rules $\mathcal{R}$ among all subsets of $B$. Furthermore, we established (in Theorem 3.15) that $K^*$ itself contains a unique minimal substate $\bigcap K^*$, and that

$$\bigcap K^* \supseteq \bigcap J_0 \supseteq \bigcup \{ \bigcap T_N^k(J_0) \mid k < \omega \}.$$ 

As a corollary, we showed that every successful query 'P(x)' in a system of rules including negations and embedded implications has a definite answer substitution for the variables x. However, we also demonstrated by example that the transformation $T(J) = T_N(J) \cap J_0$ is not, in general, continuous, and that the inequality in Theorem 3.15 is not, in general, an equality. As a result, despite the similarity in the fixed-point semantics, we showed that it is not possible to extend the simple constructive proof of the completeness theorem for Horn-clause refutation in [29] and [1] to cover the general case of negations and embedded implications. For the details of this analysis, the reader should consult the original article.

In this, the second paper, we will present our positive results on the proof theory for clausal intuitionistic logic. Specifically, we will investigate a proposed proof procedure for negations and embedded implications which resembles the standard Horn-clause proof procedure in the following way: The proof begins with a query 'P(x)' at the top node of a refutation tree, and, if it is successful, it concludes by returning an answer substitution for the variables x. But the refutation tree in our proof appears inside a structure called an initial tableau, and whenever the proof procedure encounters a negation rule in the form $P \Leftarrow \neg Q$, or an embedded implication rule in the form $P \Leftarrow [Q \Rightarrow R]$, it creates a new structure called an auxiliary tableau which contains additional formulae and additional refutation trees. We will show that this procedure is sound (Theorem 4.3) and complete (Theorem 4.6) with respect to our fixed-point semantics, so that it faithfully computes the answers that are entailed by an intuitionistic interpretation of the negation and embedded implication rules. However, as suggested by the negative results in the first paper of this pair, the proof of the completeness theorem here is much more complicated than the proof of the completeness theorem in the pure Horn-clause case [29, 1].

The proof theory for clausal intuitionistic logic is discussed in Section 4 below, which constitutes the bulk of this paper. Section 5 contains a brief discussion of related work, and Section 6 outlines several extensions of our system that are left open for future investigation.

4. NEGATIONS AND EMBEDDED IMPLICATIONS: INFERENCE

In this section, we will analyze a proposed proof procedure for a system of rules including negations and embedded implications. We will call it a tableau proof procedure because it resembles in some respects the tableau procedure used by Saul
Kripke in his original completeness theorem for modal logic [13–15]. In particular, there is a direct correspondence in our system between the tableaux constructed in a proof and the lattice of substates defined in the semantics, as there is in Kripke's system. However, instead of building the tableaux out of sets of first-order formulae, as Kripke did, generalizing the semantic tableaux of Beth, we will build them out of Horn-clause refutation trees. Our proposed proof procedure is therefore a natural generalization of Horn-clause logic programming. A similar approach has also been suggested by Graham Wrightson in [31].

Actually, we will discuss two slightly different proof procedures in this section, and several additional variations. The first procedure, called the standard proof procedure, is intended for most practical computations, but it is difficult to work with directly in the soundness and completeness theorems. This procedure will be described informally in Section 4.1, and it will be illustrated with two detailed examples. The second procedure, called the simplified proof procedure, is intended for use in the soundness and completeness theorems, but it is potentially very inefficient. This procedure will be formally specified in Section 4.2. The soundness theorem for clausal intuitionistic logic will then be proven in Section 4.3, and the completeness theorem will be proven in Section 4.4. In Section 4.5, using these results, we will return to the fixed-point semantics of Section 3 and prove a sharper version of Theorem 3.15. Finally, in Section 4.6, we will demonstrate the equivalence of the standard and the simplified proof procedures, and the equivalence of several other variations in the proof procedure that should be useful in practice.

4.1. The Standard Proof Procedure

The main components of the standard proof procedure are illustrated in Figures 1 and 2 below. Each proof begins with an initial tableau, labeled \( T_0 \), which is then extended into a system of auxiliary tableaux labeled \( T_1, T_2, \) etc. Each tableau is divided into at most four sections: (1) a section at the top containing one, or possibly two, principal refutation trees; (2) an optional section containing a set of subordinate refutation trees; (3) an optional rule base containing a set of rules; and (4) a data base containing a set of atomic formulae. A refutation tree in either of the top two sections of the tableau is a simple And/Or refutation tree [24, 25] as described in Section 2 (Paper I), but some of the variables in these trees have special prefixes. A variable with a prefix '!' is called an existential variable, and a variable with a prefix '?' is called a local tableau variable. At the beginning of the proof, the initial tableau will have the query \( P(x) \) at the top of its principal refutation tree and the substate \( s_0 \) in its data base, but nothing more. As the proof proceeds, the refutation trees will be extended, the various auxiliary tableaux will be constructed, and the answer substitutions will be computed and returned to the appropriate nodes in the trees. An individual refutation tree will close with an answer substitution at its topmost node, exactly as it does in the Horn-clause case, but in addition an auxiliary tableau will close with an answer substitution for its local variables \(?x\) whenever the principal refutation tree in the tableau closes. Finally, if the principal refutation tree in the initial tableau \( T_0 \) closes with an answer substitution \( \sigma \), we will say that the entire system of tableaux is closed with answer substitution \( \sigma \), and the proof will be complete.

The key point in understanding these proofs is to understand the construction of the auxiliary tableaux, and the role of the special prefixed variables. The basic
procedure is this: Whenever a node in one of the refutation trees unifies with the left-hand side of a rule in the form $P \leftarrow Q$ or $P \leftarrow [Q \Rightarrow R]$, we apply the unifying substitution $\sigma$ to the right-hand side of the rule and we construct a new auxiliary tableau with the formula $Q(x; y)\sigma$ added to its data base. In this step, all free variables from the left-hand side of the rule are given the prefix '?', and all variables that appear only on the right-hand side of the rule are given the prefix '!'. Then, in all subsequent unification steps, these prefixed variables must be treated specially. The existential variables '!y' are treated as constants, and cannot be bound to any other term. The local tableau variables '?x' are more complex. These variables are treated as true variables in all unification steps, but they are never renamed, and they are subject to two restrictions:

1. All substitutions for the variables '?x' must be carried down along the branches of the refutation tree, and all subsequent bindings for these variables must be consistent with the original substitutions. This means that we always compute the unifying composition [28] [4] with respect to the local tableau variables.

2. A local tableau variable '?x_i' may not be bound to a term containing an existential variable '!y_j' that was created in the same tableau in which '?x_i' was created, or in a higher level tableau.

This last restriction is similar to the usual restrictions on "eigenvariables" in natural deduction proofs, and closely related to the use of Skolem functions in resolution proofs (for example, see [3, Chapter IV.8]).

A final rule governs the closure of the auxiliary tableaux: Whenever one of the principal refutation trees in an auxiliary tableau closes with an answer substitution at its topmost node, we collect all of the bindings for the local tableau variables that are included in this answer substitution, we strip the '?' prefixes from those variables that were originally created in that particular auxiliary tableau, and we return the resulting substitution to the node in the lower-level tableau from which the auxiliary tableau was generated. From then on, the stripped variables are treated as ordinary variables, without special prefixes.

For a better understanding of these rules, we will analyze two concrete examples of the standard proof procedure in operation. The rationale for the various steps in the procedure will be explained as the proof is constructed.

Example 4.1. This example is the same as Example 3.17 in Section 3.4, except that we will now develop systematically a proof that $P_1(a)$ is true in all substates that satisfy the rules. Assume that $\mathcal{R}$ consists of the following:

$$P_1(x) \leftarrow -Q_1(x, y),$$

$$P_2(x) \leftarrow -Q_2(x, y),$$

$$Q_1(x, y) \leftarrow -R(x, y, z),$$

$$Q_2(x, y) \leftarrow -R(x, y, z),$$

and assume that the initial substate $s_0 = \{P_2(a)\}$. Let $P_1(x_0)$ be the query. We would expect the proof to return an answer substitution $\sigma: \{a \leftarrow x_0\}$, which it does. The standard proof procedure for this example is illustrated in Figure 1.
The proof begins with the tableau $\mathcal{T}_0$. This tableau initially contains a principal refutation tree consisting of a single node $'P_1(x_0)'$, and a data base consisting of a single ground instance $'P_2(a)'$. Normally, the principal refutation tree would be extended using rules of the form $P \equiv \land Q_j$ and $P \equiv \lor Q_j$, as indicated schematically in Figure 1. However, in this case the node containing $'P_1(x_0)'$ unifies immediately with the left-hand side of the rule (1), producing a unifying substitution $\sigma : \{x_1 \leftarrow x_0\}$, and so the auxiliary tableau $\mathcal{T}_1$ is immediately constructed as a successor to $\mathcal{T}_0$. 

FIGURE 1. The standard proof procedure for Example 4.1.
The reason for this is as follows: By our refutation assumption, the initial tableau $\mathcal{T}_0$ represents a substate $s \geq s_0$ such that, for every $\theta$, $P_1(x_j) \theta \notin s$. But since the rule $P_1(x_j) \Rightarrow \neg Q_1(x_1, y_1)$ holds in the substate $s$, we can see from Definition 3.1 that, for every $\theta$, there must exist some substate $s' \geq s$ and some ground substitution $\theta' \geq \theta$ such that $Q_1(x_1, y_1) \theta' \in s'$. This situation is represented by the auxiliary tableau $\mathcal{T}_1$, which has the formula $\neg Q_1(\exists x_1!y_1)'$ in its data base, and which has a principal refutation tree at the top beginning with $F$. Since the symbol $F$ stands for an absolute contradiction, the refutation assumption has now been reduced to the proposition that $\mathcal{T}_1$ is noncontradictory. Accordingly, if the refutation tree beginning with $F$ eventually closes, with some answer substitution $\sigma$, we will have a proof of the contrary proposition and a binding for the variable $\exists x_1'$. Note that the substate $s'$ and the substitution $\theta'$ depend, in the refutation assumption, on the substitution $\theta$, and this explains the special restrictions we have imposed on the bindings of the variable $\exists x_1'$.

But how can a refutation tree beginning with $F$ ever close? To see this, we need to examine the subordinate refutation tree associated with the rule (2) in the tableau $\mathcal{T}_0$. Basically, subordinate refutation trees generate lemmas from rules of the form $P \Rightarrow \neg Q$, and they may be constructed at any time in any tableau in which they might prove useful. (There is a question of nondeterminism here, of course, which we will investigate later; see Section 4.6.) In this case, the node $P_2(x_2)$ unifies with the ground instance $P_2(a)$ in the tableau $\mathcal{T}_0$, producing the unifying substitution $\sigma : \{ a \leftarrow x_2 \}$, and so this substitution is applied to the right-hand side of (2) to create the rule $Q_2(a, y) \Rightarrow F$. This new rule is then added to the rule base of $\mathcal{T}_0$, and it is made available for the extension of any principal refutation tree in $\mathcal{T}_0$, or in any higher-level tableau, beginning with $F$. The reason for this is as follows:

When the subordinate refutation tree in $\mathcal{T}_0$ closes with the answer substitution $\sigma : \{ a \leftarrow x_2 \}$, it means that $P_2(a) \in s$, and it then follows from the rule (2) and from Definition 3.1 that $Q_2(a, y) \theta' \notin s'$ for every $s' \geq s$ and for every ground substitution $\theta'$. But this situation can be represented by putting the formula $Q_2(a, y)$ into an open node of a principal refutation tree in $\mathcal{T}_0$, and in all higher-level tableaux, and making this part of the refutation assumption. In Figure 1, the rule $Q_2(a, y) \Rightarrow F$ has been used to extend the principal refutation tree in $\mathcal{T}_1$, beginning with $F$, and so if this tree eventually closes with an answer substitution $\sigma$, we will know that there exists no substate $s' \geq s$ containing $Q_1(\exists x_1'!y_1) \sigma'$.

To see if this is the case, the same basic steps are now repeated in the tableau $\mathcal{T}_1$. The open node in the principal refutation tree containing $Q_2(a, y_2)$ unifies immediately with the left-hand side of (4), producing a unifying substitution $\sigma : \{ a \leftarrow x_3, y_3 \leftarrow y_2 \}$. Applying this substitution to the right-hand side of the rule, the proof procedure now constructs a new tableau $\mathcal{T}_2$ with a data base containing the formula $R(a, y_3, !z_3)$. As before, for every instantiation of the variable $\exists y_3$, this new tableau represents a substate $s'' \geq s'$ such that $R(a, y_3, !z_3)'$ is true in $s''$ for some instantiation of the variable $\exists y_3$. The rule (3) now generates a subordinate refutation tree in $\mathcal{T}_2$, which is treated exactly like the subordinate refutation tree in $\mathcal{T}_0$, except that there is an additional formula in the data base of $\mathcal{T}_1$ that can unify with the branches of the tree. Note that the subordinate refutation tree associated with the rule (3) could also have been constructed in the tableau $\mathcal{T}_0$, except that the construction would not have closed there. In this case, however, the node of the tree containing $Q_1(x_4, y_4)'$ unifies immediately with the formula $Q_1(x_1, y_1)'$ in the
data base of $\mathcal{T}_1$, producing an answer substitution $\sigma: \{?x_1 \leftarrow x_4, !y_1 \leftarrow y_4\}$. Applying this answer substitution to the right-hand side of (3), the proof procedure constructs a new rule $R(?x_1, !y_1, z) \Rightarrow F'$ which is then added to the rule base of $\mathcal{T}_1$. The meaning of this rule is similar to the meaning of the rule in the rule base of $\mathcal{T}_0$, but more complex. Whatever value has been assigned to $?x_1$ and $!y_1$ in the substate $s'$, the rule says, $R(?x_1, !y_1, z)$ is not true for every instantiation of the variable $z$ in every substate $s'' \supseteq s'$.

Next, continuing the proof procedure in the tableau $\mathcal{T}_2$, we can see that the refutation trees close properly at this level, and that the correct answer substitutions are computed for the local tableau variables $?x_1$ and $?y_3$'. The principal refutation tree in $\mathcal{T}_2$ beginning with $F$ includes the open node $R(?x_1, !y_1, z_4)$, and this node unifies immediately with the formula $R(a, ?y_3, !z_2)$ in the data base, producing the answer substitution $\sigma: \{a \leftarrow ?x_1, !y_1 \leftarrow ?y_3, !z_2 \leftarrow z_4\}$. Recall the restrictions imposed on the bindings of the local tableau variables $?x_1$ and $?y_3$, First, the unification at this node must be a unifying composition with respect to the local tableau variables, but this causes no problems in the present case because $?x_1$ and $?y_3$ have not yet acquired any bindings. Second, the variable $?y_3$ must not be bound to an existential variable that was created in the tableau $\mathcal{T}_2$, or in any higher-level tableau, but this causes no problems in the present case because the variable $!y_1$ was created in $\mathcal{T}_1$. Thus the principal refutation tree closes, correctly, with the answer substitution $\sigma: \{a \leftarrow ?x_1, !y_1 \leftarrow ?y_3, !z_2 \leftarrow z_4\}$. This means that $\mathcal{T}_2$ itself closes with an answer substitution for the local tableau variables $?x_1$ and $?y_3$, as indicated in Figure 1.

Finally, since $?x_1$ and $?y_3$ were originally created in $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively, the answer substitution for these variables must be returned to the nodes that generated them. First, the prefix '?' is stripped from the variable $?y_3$. The resulting substitution $\sigma: \{a \leftarrow ?x_1, !y_1 \leftarrow y_3\}$ is then returned to the node containing $Q_2(a, y_2')$ and composed with the unifying substitution at that node, producing the answer substitution $\sigma: \{a \leftarrow ?x_1, !y_1 \leftarrow y_2\}$. This substitution is then passed up to the top of the principal refutation tree in $\mathcal{T}_1$. But this means that the tableau $\mathcal{T}_1$ itself is now closed, and that the answer substitution $\sigma: \{a \leftarrow ?x_1\}$ must likewise be returned to the node in $\mathcal{T}_0$ that generated it. Again, the prefix '?' is stripped from the variable $?x_1$, and the resulting substitution is composed with the unifying substitution $\sigma: \{x_1 \leftarrow x_0\}$. At this point, though, the principal refutation tree in $\mathcal{T}_0$ closes, too, with the answer substitution $\sigma: \{a \leftarrow x_0\}$, and we finally receive a response to our query. Putting everything together, we have constructed a proof of the ground atomic formula $P_1(a)$ from $s_0$ and $R$.

**Example 4.2.** We now consider a slightly more complex example, which includes function symbols and which involves a combination of various types of rules. Assume that $R$ consists of the following:

$$P_1(x) \equiv [Q_1(x, y) \Rightarrow R(x, g(y))],$$

$$P_2(x) = -Q_2(g(x), f(y)),$$

$$Q_2(x, y) \equiv [Q_3(x, y) \Rightarrow R(x, y)],$$

$$R(x, g(y)) \equiv Q_1(f(z), y) \land Q_2(z, x),$$

and assume that the initial substate $s_0 = \{P_2(a)\}$. It turns out that the ground
instance \(P_1(f(g(a)))\) is entailed by \(s_0\) and \(R\) in this case. In fact, if we apply the transformation \(T_{N^*}\) successively to \(J_0\), as we did in the examples in Section 3.4, we can show that \(\square T_{N^*}(J_0)\) converges after two iterations to

\[\square \mathcal{K}^* = \square T_{N^*}(J_0) = \{ P_2(a), P_1(f(g(a)))\}.

Thus, the query \(P_1(x,)?\) ought to succeed with an answer substitution \(\sigma : \{ f(g(a)) \leftarrow x_0\}\). Let us see if this is the case.

The standard proof procedure for Example 4.2 is illustrated in Figure 2. The tableau \(T_0\) initially contains a principal refutation tree consisting of a single node \(P_1(x_0)\), and a data base consisting of a single ground instance \(P_2(a)\). However, the node containing \(P_1(x_0)\) unifies immediately with the left-hand side of (5), producing a unifying substitution \(\sigma : \{ x_1 \leftarrow x_0\}\), and so the auxiliary tableau \(T_1\) is immediately constructed as a successor to \(T_0\). The reason for this is as follows: By our refutation assumption, the initial tableau \(T_0\) represents a substate \(s \supseteq s_0\) such that, for every \(\theta\), \(P_1(x_1) \theta \not\in \mathcal{S}\). But since the rule \(P_1(x_1) \Rightarrow \{ Q_1(x_1, y_1) \leftarrow R(x_1, g(y_1))\}\) holds in the substate \(s\), we can see from Definition 3.2 that, for every \(\theta\), there must exist some substate \(s' \supseteq s\) and some ground substitution \(\theta' \supseteq \theta\) such that \(Q_1(x_1, y_1) \theta' \in s'\) and \(R(x_1, g(y_1)) \theta' \in s'\). Up to this point, the analysis is similar to the analysis of the negation rule in Example 4.1, but there are now two ways for the tableau \(T_1\) to close. First, the principal refutation tree in \(T_1\) beginning with \(\mathcal{F}\) could close, exactly as it did in Example 4.1 (Note: This tree has been omitted from Figure 2 because of a lack of space.) Second the principal refutation tree in \(T_1\) beginning with \(\{ R(?x_1, g(!y_1))\}\) could close, with some answer substitution \(\sigma\) for the variable \(?x_1\). In this latter case, we would know that any substate \(s' \supseteq s\) containing \(Q_1(?x_1, y_1)\sigma\) must also contain \(R(?x_1, g(!y_1))\sigma\), and thus \(P(?x_1)\sigma\) would be true in \(s\) by the rule (5).

To see if this is the case, the proof procedure now continues in the tableau \(T_1\). Here, the node containing \(\{ R(?x_1, g(!y_1))\}\) unifies immediately with the left-hand side of (8), producing the unifying substitution \(\sigma : \{ ?x_1 \leftarrow x_1, !y_1 \leftarrow y_1\}\). The rule (8) is an ordinary Horn clause, of course, so the substitution is applied to each of the conjuncts and the proof continues. (Recall that we are using And/Or refutation trees, by convention, instead of SLD refutation trees.) The left conjunct unifies immediately with the formula in the data base of \(T_1\), producing the unifying substitution \(\sigma : \{ f(z_1) \leftarrow ?x_1\}\), and the right conjunct unifies with the left-hand side of (7), producing the unifying substitution \(\sigma : \{ z_1 \leftarrow x_4, ?x_1 \leftarrow y_4\}\). This means that the proof procedure must now construct another auxiliary tableau \(T_2\). Applying the substitution \(\sigma : \{ z_1 \leftarrow x_4, ?x_1 \leftarrow y_4\}\) to the right-hand side of (7), the embedded implication becomes \(Q_3(z_1, ?x_1) \Rightarrow R(z_1, ?x_1)\), and so the free variable \('z_1'\) is converted into a new local tableau variable \('?z_1'\). Thus the new tableau \(T_2\) has the formula \(Q_3(?z_1, ?x_1)\) in its data base, and it contains two principal refutation trees: a refutation tree beginning with the node \(R(?z_1, ?x_1)\), and a refutation tree beginning with \(\mathcal{F}\).

Now one way to continue the proof in the tableau \(T_2\) is to expand the principal refutation tree beginning with \(\{ R(?z_1, ?x_1)\}\). But another way to continue the proof is to expand the principal refutation tree beginning with \(\mathcal{F}\). Where does this tree come from? To see this, we have to return to the tableau \(T_0\), where a subordinate refutation tree has been constructed for the rule (6). Here the analysis is identical to the analysis in Example 4.1. The node \(P_2(x_2)\) unifies with the ground instance
FIGURE 2. The standard proof procedure for Example 4.2.

'P_2(a)' in T_0, producing the unifying substitution \( \sigma : \{ a \leftarrow x_2 \} \), and so this substitution is applied to the right-hand side of (6) to create the new rule \( Q_3(g(a), f(y)) \rightarrow \text{F} \). This new rule is then added to the rule base of \( T_0 \), and it is made available for the extension of any principal refutation tree in \( T_0 \), or in any higher-level tableau, beginning with \( \text{F} \). The reason for this is as follows: When the subordinate refutation tree closes in \( T_0 \) with the answer substitution \( \sigma : \{ a \leftarrow x_2 \} \), it means that \( P_2(a) \in s \), and it then follows from (6) and from Definition 3.1 that \( Q_3(g(a), f(y)) \theta' \notin s' \) for every \( s' \geq s \) and for every ground substitution \( \theta' \).
In the present proof, in fact, the tableau $T_2$ closes precisely because of the rule $\text{Q}_2(g(a), f(y)) \Rightarrow F$. The formula $\text{Q}_2(g(a), f(y))$ unifies immediately with the data base of $T_2$, producing the unifying substitution $\sigma: \{ g(a) \leftarrow z_1, f(y_2) \leftarrow x_1 \}$, and thus the principal refutation tree beginning with $F$ is closed. The bindings of the local tableau variables are now collected in $T_2$, and the proof procedure begins to return them to the lower-level tableaux. Since $?z_1$ was created in $T_2$, its prefix is stripped away as the answer substitution is returned from $T_2$ to $T_1$, but the prefix on the variable $?x_1$ is retained. Back in the tableau $T_1$, the answer substitution $\sigma: \{ g(a) \leftarrow z_1, f(y_2) \leftarrow x_1 \}$ arrives at the ‘And’ node that was originally created by the rule (8), and the proof procedure computes its unifying composition with the substitution $\sigma: \{ f(z_1) \leftarrow x_1 \}$. The answer is $\sigma: \{ f(g(a)) \leftarrow x_1, g(a) \leftarrow z_1, g(a) \leftarrow y_2 \}$. Since the principal refutation tree in the tableau $T_1$ has now closed, the proof procedure returns the answer substitution $\sigma: \{ f(g(a)) \leftarrow x_1 \}$ to the tableau $T_n$, stripping away the last remaining ‘?’ prefix in the process. It is now obvious that the principal refutation tree in $T_0$ will also close, with answer substitution $\sigma: \{ f(g(a)) \leftarrow x_0 \}$, and thus the proof is complete.

It is instructive to examine the one remaining open tree in this proof, to see why it does not also lead to closure. Notice that the node $R(?z_1, ?x_1)$ in the principal refutation tree in the tableau $T_2$ unifies with the left-hand side of (8), producing an ‘And’ node. Now it might appear that the left conjunct $Q_1(f(z_2), y_3)$ should unify with the formula $Q_1(?x_1, y_1)$ in the data base of the tableau $T_1$. However, this unification would violate our restrictions on the bindings of the variable $?x_1$ for two reasons: First, the prior substitution $\sigma: \{ g(y_3) \leftarrow x_1 \}$ would be carried down to this conjunct, and the original binding for $?x_1$ would then conflict with the substitution $\sigma: \{ f(z_2) \leftarrow x_1 \}$ which otherwise results from the unification step. Second, if $y_3$ were bound to ‘$y_1$’ as a result of the unification step, then ‘$?x_1$’ would be bound to ‘$g(y_1)$’, which is prohibited. The simplest way to enforce these restrictions is to write down the terms and the prior substitution as follows:

$$[Q_1(f(z_2), y_3), ?x_1],$$
$$[Q_1(?x_1, y_1), g(y_3)],$$

and then attempt to unify the bracketed expressions. In this case, the attempted unification fails. Hopefully, an implementation of our proof procedure would detect this failure early, because otherwise, as Figure 2 indicates, the right conjunct in the tableau $T_2$ could very well generate an infinite sequence of auxiliary tableaux. The lesson here, as in the Horn-clause case, is that the mere existence of a closed system of tableaux does not guarantee that an arbitrary ordering of the proof procedure will terminate in finite time.

4.2. The Simplified Proof Procedure

Although the standard proof procedure is intended for most practical computations in clausal intuitionistic logic, it is awkward to use the procedure directly in the proofs of the soundness and completeness theorems. We will thus introduce in this section a simplified proof procedure, with a greater degree of homogeneity, which is easier to analyze.
There are two main differences between the standard and the simplified proof procedures: First, in the simplified proof procedure, the negation rules $P \Leftarrow \neg Q$ generate principal refutation trees directly, as if they were Horn clauses in the form $F \Leftarrow P(x) \land Q(x; y)$. Thus, there are no subordinate refutation trees, and there is no rule base, in a simplified tableau. However, the principal refutation trees beginning with $F$ can be constructed at any time, in any tableau, for any negation rule $P \Leftarrow \neg Q$, and they therefore tend to proliferate wildly. Second, in the simplified proof procedure, an auxiliary tableau $\mathcal{T}_n'$ can be constructed for any rule $P \Leftarrow Q$ or $P \Leftarrow [Q \Rightarrow R]$, at any time, even if the formula $P(x)$ on the left-hand side of the rule has not yet unified with a node $N$ in one of the refutation trees in $\mathcal{T}_n$. Thus, instead of attaching an auxiliary tableau to a particular node $N$, with a particular unifying substitution, as in the standard proof procedure, the simplified proof procedure attaches the auxiliary tableau directly to the base tableau $\mathcal{T}_n$, and returns a general answer formula when the auxiliary tableau closes. Again, this procedure could easily lead to a wild proliferation of auxiliary tableaux.

For an illustration of these differences, the simplified proof procedure for Example 4.1 is shown in Figure 3. The reader should compare this figure with Figure 1. The two subordinate refutation trees in Figure 1 have been replaced in Figure 3 by the two principal refutation trees for the rules $F \Leftarrow p_2(x_2) \land q_2(x_2, y_2)$ and $F \Leftarrow q_1(x_4, y_4) \land r(x_4, y_4, z_4)$. Also, the tableau $\mathcal{T}_2$ is associated with the generic rule $q_2(x_3, y_3) \Leftarrow r(x_3, y_3, z_3)$ in Figure 3, instead of being specialized by the unifying substitution $\sigma: \{a \leftarrow x_1\}$, as it was in Figure 1. Thus, when $\mathcal{T}_2$ closes in Figure 3, the answer formula $\{q_2(x_1, y_1)\}$ is returned to $\mathcal{T}_1$, where it subsequently unifies with the formula $\{q_2(x_2, y_2)\}$. The answer substitution $\sigma: \{a \leftarrow ?x_1\}$ does not appear in Figure 3 until the unifying composition is computed for the ‘And’ node in the principal refutation tree in $\mathcal{T}_1$. Figure 3 also shows the many superfluous auxiliary tableaux that can be generated by the simplified proof procedure. For example, $\mathcal{T}_3$ is constructed as an auxiliary tableau to $\mathcal{T}_0$, but it never closes, and, even if it did close, its answer formula would never unify with the nodes of the principal refutation tree in $\mathcal{T}_0$.

Despite these practical complications, the simplified proof procedure is a much more tractable theoretical construct than the standard proof procedure. We will now state rigorously the rules governing the construction of these proofs. At each step, the proof procedure identifies (nondeterministically) a tableau $\mathcal{T}$ in the current system of tableaux, and it executes (nondeterministically) one of the applicable rules to construct an updated system of tableaux. Recall that there are five kinds of expressions in our language: $P \Leftarrow \land Q_j$, $P \Leftarrow \lor Q_j$, $P \Rightarrow Q$, $P \Leftarrow \neg Q$, and $P \Leftarrow [Q \Rightarrow R]$. Each of these expressions has associated with it a tableau extension step and a tableau closure step, as follows:

Rules $P \Leftarrow \land Q_j$ and $P \Leftarrow \lor Q_j$:

**Extension:** If any node $N$ in any refutation tree in the tableau $\mathcal{T}$ unifies with the left-hand side $P(x)$ of any rule in the form $P \Leftarrow \land Q_j$ or $P \Leftarrow \lor Q_j$, extend the node $N$ with an ‘And’ branch or an ‘Or’ branch, accordingly, as in the pure Horn-clause case.

**Closure:** If any node $N$ in any refutation tree in the tableau $\mathcal{T}$ unifies with an atomic formula in the data base of $\mathcal{T}$, or in the data base of any predecessor of $\mathcal{T}$, add the unifying substitution to the node $N$ as one of its
answer substitutions. Similarly, if any node \( \mathcal{N} \) in any refutation tree in the tableau \( \mathcal{T} \) unifies with an answer formula (see below) returned from an auxiliary tableau attached to \( \mathcal{T} \), add the unifying substitution to the node \( \mathcal{N} \) as one of its answer substitutions. For nodes that have been extended into 'And' branches or 'Or' branches, propagate the answer substitutions back up the refutation tree exactly as in the pure Horn-clause case, using unifying compositions as necessary.

FIGURE 3. The simplified proof procedure for Example 4.1.
Rules $P \Rightarrow \neg Q$:

**Extension:** For any rule in the form $P(x) \Rightarrow \neg Q(x; y)$ that has not yet been extended in the tableau $\mathcal{T}$, construct a principal refutation tree in $\mathcal{T}$ beginning with $F$ and extend it to an ‘And’ branch including the formulae $P(x)$ and $Q(x; y)$. In this construction, the variables $x$ and $y$ must be uniquely renamed.

**Closure:** Whenever an answer substitution $\sigma$ is propagated back to a node containing $F$ in tableau $\mathcal{T}$, we will say that the principal refutation tree in $\mathcal{T}$ beginning with $F$ has closed with answer substitution $\sigma$.

Rules $P \Leftarrow \neg Q$ and $P \Leftarrow [Q \Rightarrow R]$:

**Extension:** For any rule in the form $P(x) \Leftarrow \neg Q(x; y)$ or $P(x) \Leftarrow [Q(x; y) \Rightarrow R(x; y)]$ that has not yet been extended in the tableau $\mathcal{T}$, construct a new auxiliary tableau $\mathcal{T}'$ and attach it to $\mathcal{T}$. In this construction, the variables $x$ and $y$ must be uniquely renamed. Consider the atomic formula $Q(x; y)$ from the right-hand side of the rule. Proceed as follows: Replace every variable ‘$y$’ with an existential variable ‘!$y$’, replace every variable ‘$x$’ with a local tableau variable ‘?$x$’, and add the modified formula $Q(?x; !y)$ to the data base of $\mathcal{T}'$. In the case of a rule $P \Leftarrow [Q \Rightarrow R]$, replace the variables of $R(x; y)$ in the same way, and construct a principal refutation tree in $\mathcal{T}'$ with the modified formula $R(?x; !y)$ at its topmost node.

**Closure:** Notice that the principal refutation trees in the auxiliary tableau $\mathcal{T}'$ come in two varieties: Some begin with $R(?x; !y)$, and some begin with $F$. But whenever a principal refutation tree of either variety closes with an answer substitution $\sigma'$, we will say that the auxiliary tableau $\mathcal{T}'$ is closed with answer substitution $\sigma'$. The substitution $\sigma'$ may contain $?x$ variables that were created in $\mathcal{T}'$, as well as $?x$ variables that were created in $\mathcal{T}$ and its predecessors, as well as ordinary variables without prefixes. Proceed as follows: Strip the ‘?’ prefix from all the $?x$ variables that were created in $\mathcal{T}'$, and denote the resulting substitution by $\sigma$. If $P(x)$ is the atomic formula on the left-hand side of the rule that generated the auxiliary tableau $\mathcal{T}'$, we will refer to $P(x)\sigma$ as the answer formula returned from the tableau $\mathcal{T}'$. Finally, if $\mathcal{N}$ is any node in any refutation tree in $\mathcal{T}$ that unifies with $P(x)\sigma$, add the unifying substitution to $\mathcal{N}$ as one of its answer substitutions.

In all of these steps, of course, the variables $?x$ and !$y$ are treated specially. The !$y$ variables are treated as constants, and the $?x$ variables are subject to the following two restrictions:

1. All substitutions for the $?x$ variables must be carried down along the branches of the refutation trees, and every unification step in the proof must be a unifying composition with respect to these variables.

2. A local tableau variable ‘?$x$’ may not be bound to a term containing an existential variable ‘!$y$’ that was created in the same tableau in which ‘?$x$’ was created, or in a higher-level tableau.

Note that these are exactly the same restrictions imposed in the standard proof procedure.
4.3. Soundness

In this subsection, we will prove the soundness theorem for clausal intuitionistic logic, assuming that all proofs are constructed according to the simplified proof procedure. As in the pure Horn-clause case, we must assume that the individual constants in the query \( P(x) \) and the rules \( \mathcal{R} \) also appear in the initial substate \( s_0 \). The soundness theorem can then be stated as a close paraphrase of Theorem 2.17.

Theorem 4.3 (Soundness). Let \( s_0 \) be a (possibly infinite) initial substate in \( B \), and let \( \mathcal{R} \) be a (possibly infinite) set of rules including negations and embedded implications. Assume that the individual constants in \( P(x) \) and \( \mathcal{R} \) appear in \( s_0 \), and assume that there exists a closed system of tableaux for \( P(x) \) with answer substitution \( \sigma \). Then \( (\exists x)P(x) \) is uniformly entailed by \( s_0 \) and \( \mathcal{R} \) with some ground substitution \( \theta \) such that \( \sigma(x) \leq \theta \).

The basic idea of the proof is to convert a closed system of tableaux into a finite sequence of applications of \( T_N \) to \( \mathcal{J}_0 \). Intuitively, a tableau \( \mathcal{T} \) that has no closed auxiliary tableaux extending from it represents the set of substates in \( \mathcal{J}_0 \), and an answer formula that is returned to the predecessor of \( \mathcal{T} \) represents a ground instance in \( \mathcal{T}|T_N(\mathcal{J}_0) \). For a concrete illustration of this correspondence between the auxiliary tableaux in a proof and the operators \( T_k \) in the semantics, the reader should compare the proof procedures in Figures 1 and 3 with the semantic analysis of the same set of rules in Example 3.17. Note that the tableau \( \mathcal{T}_2 \) in Figures 1 and 3 corresponds to the set \( \mathcal{T}_1 \), the tableau \( \mathcal{T}_1 \) corresponds to the set \( T_N(\mathcal{J}_0) \), and the tableau \( \mathcal{T}_0 \) corresponds to the set \( T_N^2(\mathcal{J}_0) \).

To formalize this idea, it will be helpful to establish some additional terminology. Let us focus on a particular closed system of tableaux for the duration of the proof. Consider any answer substitution \( \sigma \) at any node \( \mathcal{N} \) in any tableau \( \mathcal{T} \) in this closed system of tableaux. The answer substitution \( \sigma \) must have resulted from the closure of a (possibly empty) refutation tree constructed out of a finite set of rules \( P \Leftarrow \land Q_j \) and \( P \Leftarrow \lor Q_j \), including (possibly) a special rule in the form \( F \Leftarrow P(x) \land Q(x,y) \). Let us identify the specific branches of the refutation tree that produced the answer substitution \( \sigma \), and let us call this the rule dependency tree for \( \sigma \). Note that each branch of this rule dependency tree must terminate in either: (1) a ground instance in \( s_0 \), or a Horn clause \( P(x) \Leftarrow \) with a null antecedent; (2) a data-base formula \( Q(?x; ?y) \) in \( \mathcal{T} \) or one of its predecessors; or (3) an answer formula \( P(x)\sigma \) returned from one of the auxiliary tableaux attached to \( \mathcal{T} \).

In addition to the concept of a rule dependency tree, we will need the related concept of a tableau dependency tree. Consider any node \( \mathcal{N} \) in the rule dependency tree for \( \sigma \). If \( \mathcal{N} \) terminates in an answer formula returned from an auxiliary tableau \( \mathcal{T}' \), as in case (3) above, then an answer substitution at \( \mathcal{N} \) will be said to depend on the closure of the tableau \( \mathcal{T}' \), or to depend on the closure of one of the principal refutation trees in \( \mathcal{T}' \), which is an equivalent statement. Since the answer substitutions in the nodes of the principal refutation tree in \( \mathcal{T}' \) might in turn depend on the closure of further auxiliary tableaux, we can define the tableau dependency tree for \( \sigma \) in the obvious way. We will then define the degree of the answer substitution \( \sigma \) by reference to the depth of this tableau dependency tree: If the rule dependency tree for \( \sigma \) contains no nodes whose answer substitution depends on the closure of an
auxiliary tableau, then $\sigma$ has degree 0. But if $\sigma$ is an answer substitution at a node $N'$ that depends on the closure of a principal refutation tree in the auxiliary tableau $T'$, and if the degree of the answer substitution at the top of this principal refutation tree is $n$, then the degree of $\sigma$ is $n + 1$. Finally, for an arbitrary $\sigma$ at an arbitrary node $N$, we will define the degree of $\sigma$ to be equal to the maximum degree of the answer substitutions in its rule dependency tree. Since the entire system of tableaux is finite, it is obvious that the tableau dependency tree for $\sigma$ is finite, and that it must terminate in a set of rule dependency trees whose answer substitutions all have degree 0.

Working with these definitions, we will now establish a soundness lemma for tableau dependency trees. In this lemma, when we refer to an atomic formula $P_0(\overline{x}; \overline{y}; z)$ in a particular tableau $T$, we are using the variable $\overline{x}$ to denote the set of all $\overline{x}$ variables in $T$ and its predecessors, and we are using the variable $\overline{y}$ to denote the set of all $\overline{y}$ variables in $T$ and its predecessors. The new variables created in an auxiliary tableau attached to $T$ will then be denoted by $\overline{x}$ and $\overline{y}$, respectively. With this notation, the soundness lemma is stated as follows:

**Lemma 4.4.** Consider a tableau $T$, and a node $N$ in $T$ that contains an atomic formula $P_0(\overline{x}; \overline{y}; z)$ and an answer substitution $\sigma$ of degree $k$. Consider also any substate $s \in T_N^k(\emptyset)$ that includes the individual constants in $P(\overline{x})$ and $R$, and satisfies the following condition: We collect all the data-base formulae $Q(\overline{x}; \overline{y})$ in $T$ and its predecessors, we instantiate the variables $\overline{x}$ and $\overline{y}$ to any arbitrary ground terms consistent with $\sigma$, and we require that these instantiated data-base formulae be contained in the substate $s$. Let $\theta$ be any ground substitution that includes these instantiations of the variables $\overline{x}$ and $\overline{y}$, and further instantiates the remaining free variables (if any) in $P_0(\overline{x}; \overline{y}; z)\sigma$ to some ground terms in $U(s)$ consistent with $\sigma$. Then it follows that $P_0(\overline{x}; \overline{y}; z)\sigma \circ \theta \in s$.

**Proof.** The proof of Lemma 4.4 is by induction on the degree $k$. For the case $k = 0$, the rule dependency tree for $\sigma$ contains no nodes whose answer substitution depends on the closure of an auxiliary tableau, and the conclusion of the lemma follows directly from the soundness of Horn-clause refutation. We need only consider specifically the branches of the rule dependency tree that terminate in nodes containing the data base formulae $Q(\overline{x}; \overline{y})$. For these branches, by the conditions of Lemma 4.4, the variables $\overline{x}$ and $\overline{y}$ have been instantiated to some ground terms consistent with $\sigma$, and the instantiated formulae $Q(\overline{x}; \overline{y})$ have been included in the substate $s$. Since the answer substitution $\sigma$ has been computed using unifying compositions on the variables $\overline{x}$, however, we can replace the variables $\overline{x}$ and $\overline{y}$ throughout the rule dependency tree with their corresponding ground terms, and we are guaranteed that the unifications will still go through. Thus, by Lemma 2.15 in the proof of Theorem 2.17 (soundness), it follows that $P_0(\overline{x}; \overline{y}; z)\sigma \circ \theta \in s$.

Now assume that Lemma 4.4 has been established for all answer substitutions of degree less than $k$. Consider any branch of any rule dependency tree in $T$ that terminates in an answer formula $P_1(\overline{x})\sigma_1$, and assume that the substitution $\sigma_1$ results from the closure of an auxiliary tableau $T'$ with an answer substitution $\sigma'_1$ of degree less than $k$. We observe that the substitution $\sigma_1$ may contain the variables $\overline{x}$ and $\overline{y}$ from $T$ and its predecessors, as well as additional free variables $z$.
without prefixes, and that it is identical to the substitution \( \sigma'_i \) except that the prefix ‘?’ has been removed from all variables \(?x_1\) appearing in \( \sigma'_i \). (Note that the stripped variables \( x_1 \) will then be included among the free variables \( z \) in \( \sigma_i \).) Let \( s \in T^k_N(\mathcal{J}_0) \) be any substate that satisfies the conditions of Lemma 4.4. In other words, assume that \( s \) contains the data-base formulae \( Q(?x_0; !y_0) \) with the variables \(?x_0\) and \( !y_0\) instantiated to some arbitrary ground terms consistent with \( \sigma_i \). Then, if \( \theta_i \) is any further ground instantiation of the variables in \( P_i(x_1)\sigma_i \) restricted to \( U(s) \) and consistent with \( \sigma_i \), we wish to show that \( P_i(x_1)\sigma_i \circ \theta_i \in s \). Equivalently, assuming that \( \theta'_i \) is identical to \( \theta_i \) but with all its \( x_1 \) variables replaced by \(?x_1\) variables, we will show that \( P_i(?x_1)\sigma'_i \circ \theta'_i \in s \).

We will first consider the case in which the auxiliary tableau \( \mathcal{T}' \) has been constructed by the extension of a negation rule \( P_i(x_1) \leftarrow Q_1(x_1; y_1) \), so that the formula \( Q_1(?x_1; !y_1) \) has been added to the data base of \( \mathcal{T}' \). (Recall that \( x_1 \) and \( y_1 \) are distinct new variables that have not previously appeared in \( \mathcal{T} \) or its predecessors.) Since the substate \( s \in T^k_N(\mathcal{J}_0) \) is also a member of \( T^{k-1}_N(\mathcal{J}_0) \), we will use Definition 3.5 to apply the operator \( \mathcal{T}^k \) to \( T_N \) to \( T^{k-1}_N(\mathcal{J}_0) \) and thereby compute the ground instances in \( s \). In particular, we will show that the first condition of Definition 3.5 holds in \( T^{k-1}_N(\mathcal{J}_0) \) by a reductio ad absurdum: Pick any arbitrary ground substitution \( \psi \) for the variables in \( !y_1 \), and assume that there exists some \( s' \geq s \) in \( T^{k-1}_N(\mathcal{J}_0) \) such that \( Q_1(?x_1; !y_1)\sigma'_i \circ \theta'_i \circ \psi \in s' \). We will show that this assumption leads to a contradiction. The reader should note here the importance of the restrictions imposed on the substitution \( \sigma'_i \) within our unification procedure. It is only because the variables \(?x_0\) and \( ?x_1\) cannot be bound to terms containing the variables \( !y_0 \) that we can always pick a substitution \( \psi \) independently of \( \sigma'_i \circ \theta'_i \), and thus establish the first condition of Definition 3.5 for all possible instantiations of \( !y_1 \).

It should now be apparent that we can apply the induction hypothesis of Lemma 4.4 to the substate \( s' \in T^{k-1}_N(\mathcal{J}_0) \) of our reductio assumption. Since \( s' \geq s \), the ground instantiations of the data-base formulae in \( \mathcal{T} \) and its predecessors, which were assumed to be included in \( s \), must also be included in \( s' \). Since \( Q_1(?x_1; !y_1)\sigma'_i \circ \theta'_i \circ \psi \in s' \) by the reductio assumption, and since \( \sigma'_i \circ \theta'_i \circ \psi \) is consistent with \( \sigma_i \), we see that \( s' \) also includes a consistent ground instantiation of the data base formula in \( \mathcal{T}' \). Finally, we observe that the principal refutation tree in \( \mathcal{T}' \) that closed with the answer substitution \( \sigma'_i \) must have been constructed from a rule in the form \( \mathcal{F} \leftarrow P_2(x_2) \land Q_2(x_2; y_2) \), derived from a negation rule \( P_2(x_2) \leftarrow \neg Q_2(x_2; y_2) \), and thus the nodes \( P_2(x_2) \) and \( Q_2(x_2; y_2) \) must have closed with a pair of answer substitutions whose unifying composition is \( \sigma_i \). But we can now apply the induction hypothesis of Lemma 4.4 directly to each of these nodes. It is then easy to see, under all of these assumptions, that there must exist some ground substitution \( \theta_2 \) such that \( P_2(x_2)\theta_2 \) and \( Q_2(x_2; y_2)\theta_2 \) are both contained in the substate \( s' \), which contradicts the fact that the set \( T^{k-1}_N(\mathcal{J}_0) \) satisfies the rule \( P_2(x_2) \leftarrow \neg Q_2(x_2; y_2) \). Our reductio argument is now complete, and we conclude that the first condition of Definition 3.5 holds for the substate \( s \) in \( T^{k-1}_N(\mathcal{J}_0) \). Since \( s \) is preserved by the application of \( \mathcal{T}_N \) to \( T^{k-1}_N(\mathcal{J}_0) \), it follows that \( P_i(?x_1)\sigma'_i \circ \theta'_i \in s \).

We will now consider the case in which the auxiliary tableau \( \mathcal{T}' \) has been constructed by the extension of some embedded implication rule \( P_i(x_1) \leftarrow [Q_1(x_1; y_1) \Rightarrow R_1(x_1; y_1)] \), so that \( \mathcal{T}' \) includes a principal refutation tree beginning with the formula \( R_1(?x_1; !y_1) \). The analysis here is very similar to the analysis of the
negation rule, except that we must now show that the second condition of Definition 3.5 holds for \( s \) considered as a substate of \( T_{k-1}^{(J)}(\delta) \). If the answer substitution \( \sigma' \) resulted from the closure of the principal refutation tree in \( \mathcal{F}' \) beginning with the formula \( R_i(?x_1; ?y_1) \), then this conclusion follows directly from our induction hypothesis. For if \( \theta_1' \) is a ground substitution for the variables in \( P_i(?x_1)\sigma_1' \) restricted to the terms in \( U(s) \), and if \( \psi \) is an arbitrary ground substitution for the variables in \( !y_1 \), and if \( \sigma_1' \circ \theta_1' \circ \psi \in s' \) for some \( s' \geq s \) in \( T_{k-1}^{(J)}(\delta) \), then the substate \( s' \) satisfies the conditions of the induction hypothesis and it follows immediately that \( R_i(?x_1; ?y_1)\sigma_1' \circ \theta_1' \circ \psi \in s' \). On the other hand, if the answer substitution \( \sigma' \) resulted from the closure of a principal refutation tree in \( \mathcal{F}' \) beginning with \( F \), which is also possible, then the second condition of Definition 3.5 would hold for \( s \) in \( T_{k-1}^{(J)}(\delta) \) by virtue of the same reductio argument as before. In either case, we would conclude that \( P_i(?x_1)\sigma_1' \circ \theta_1' \in s \).

We have now established the conclusion of Lemma 4.4 for any node \( \mathcal{N} \) in \( \mathcal{F} \) that terminates in an answer formula \( P_i(?x_1)\sigma_1 \) of degree \( k \). For the final step of the proof, consider any arbitrary node \( \mathcal{N} \) in \( \mathcal{F} \) that contains a formula \( P_i(?x_1; ?y_1; z) \) and an answer substitution \( \sigma \) of degree \( k \). Assume that the conditions of Lemma 4.4 are satisfied for some substate \( s \in T_{k}^{(J)}(\delta) \). Because of the preceding arguments, any branch of the rule dependency tree for \( \sigma \) that terminates in an answer formula \( P_i(?x_1)\sigma_1 \) can now be analyzed in the same way as a branch that terminates in a data-base formula \( Q(?x_0; ?y_0) \). In particular, since the answer substitution \( \sigma \) has been computed using unifying compositions on the variables \( ?x_0 \), we can replace the variables \( ?x_0 \) and \( !y_0 \) throughout the rule dependency tree with their corresponding ground terms, and we are guaranteed that the unifications will still go through. Thus, by Lemma 2.15 in the proof of Theorem 2.17 (soundness), it follows that \( P_0(?x_0; ?y_0; z)\sigma \circ \theta \in s \). This completes the proof of Lemma 4.4. □

With this lemma in hand, the proof of the main theorem is simple. Suppose, first, that our system of tableaux is closed because the principal refutation tree for \( P(x) \) has closed with an answer substitution \( \sigma \) in the tableau \( \mathcal{F}_0 \). The substitution \( \sigma \) must have some finite degree: call it \( m \). Since there are no variables of the form \( ?x \) or \( !y \) in \( \mathcal{F}_0 \), and since the data base of \( \mathcal{F}_0 \) consists solely of the ground instances in the initial substate \( \delta_0 \), Lemma 4.4 reduces to a simple statement: For every substate \( s \in T_{m}^{(J)}(\delta) \) that includes the individual constants in \( P(x) \) and \( \delta \), and for every ground substitution \( \theta \) for the variables in \( P(x)\sigma \) restricted to the terms in \( U(s) \), \( P(x)\sigma \circ \theta \in s \). Suppose, on the other hand, that our system of tableaux is closed because some principal refutation tree in the form \( F \leftarrow P \land Q \) has closed with an answer substitution \( \sigma \) of degree \( m \) in the tableau \( \mathcal{F}_0 \). In this case, which is the degenerate case of an inconsistent set of rules, we can show by the same analysis used in our reductio argument above that there exists no substate \( s \) in \( T_{m}^{(J)}(\delta) \). We thus arrive, trivially, at the identical conclusion: For every \( s \in T_{m}^{(J)}(\delta) \), \( P(x)\sigma \circ \theta \in s \). In either case, since \( \mathcal{K}^* \subseteq T_{m}^{(J)}(\delta) \), we have established the following result, which parallels Lemma 2.15 in the Horn-clause case:

**Lemma 4.5.** Let \( s_0 \) be a (possibly infinite) initial substate in \( \mathcal{B} \), and let \( \mathcal{R} \) be a (possibly infinite) set of rules including negations and embedded implications. Assume there exists a closed system of tableaux for \( P(x) \) with answer substitution \( \sigma \). Then for every substate \( s \in \mathcal{K}^* \) that includes the individual constants in \( P(x) \) and \( \mathcal{R} \),
and for every ground substitution \( \theta \) for the free variables in \( P(x) \sigma \) restricted to the terms in \( U(s) \), it follows that \( P(x) \sigma \circ \theta \in s \).

Theorem 4.3 now follows from Lemma 4.5, exactly as in the Horn-clause case. Assume that the initial substate \( s_0 \) includes all the individual constants in \( P(x) \) and \( \mathcal{R} \). If \( P(x) \sigma \) is a ground atomic formula, then Lemma 4.5 tells us that \( P(x) \sigma \in \bigcap \mathcal{K}^* \). Thus \( (\exists x)P(x) \) is uniformly entailed by \( s_0 \) and \( \mathcal{R} \) with ground substitution \( \sigma(x) \).

Otherwise, if \( \theta \) is any ground substitution that binds the free variables in \( P(x) \sigma \) to the ground terms in \( U(\bigcap \mathcal{K}^*) \), then Lemma 4.5 tells us that \( P(x) \sigma \circ \theta \in \bigcap \mathcal{K}^* \). Thus \( (\exists x)P(x) \) is uniformly entailed by \( s_0 \) and \( \mathcal{R} \) with ground substitution \( \sigma(x) \circ \theta \). This completes the proof of Theorem 4.3.

4.4. Completeness

In this subsection, we will prove the completeness theorem for clausal intuitionistic logic, assuming again that all proofs are constructed according to the simplified proof procedure. For simplicity, we will assume in our statement of the theorem that \( s_0 \) and \( \mathcal{R} \) are finite, but we will show how to remove this restriction in Section 4.5. Otherwise, the completeness theorem can be stated as a close paraphrase of Theorem 2.14 in the pure Horn-clause case.

Theorem 4.6 (Completeness). Let \( s_0 \) be a finite initial substate in \( \mathcal{B} \), and let \( \mathcal{R} \) be a finite set of rules including negations and embedded implications. Assume that \( (\exists x)P(x) \) is uniformly entailed by \( s_0 \) and \( \mathcal{R} \) with ground substitution \( \theta \). Then there exists a closed system of tableaux for \( P(x) \) with answer substitution \( \sigma \) such that \( \sigma(x) \leq \theta \).

We will show the contrapositive. We thus let \( \mathcal{T}_0 \) be an initial tableau for \( P(x) \), and we assume that there exists no sequence of rule applications starting with \( \mathcal{T}_0 \) that results in a closed system of tableaux with an answer substitution \( \sigma \) such that \( \sigma(x) \leq \theta \). Our objective is to construct a substate \( s \in \mathcal{K}^* \) and to show that \( P(x) \theta \notin s \).

Under this assumption, let us consider a sequence of rule applications that is fair in the following sense:

Definition 4.7. A sequence of rule applications is fair if, at every step in the sequence, any rule that is applicable at that step will be executed after only a finite number of additional steps.

It is easy to see that a fair sequence of rule applications exists. For example, since the execution of each rule creates at most a finite set of applicable new rules, any simple "first in, first out" queueing discipline will suffice. The sequence of systems of tableaux generated by these rules will be infinite, in general, and we will proceed upon this assumption. Notice that it is only the extension rules for \( P \Leftarrow \neg Q \) and \( P \Leftarrow [Q \Rightarrow R] \) which create new auxiliary tableaux, and that all other rules simply augment the existing tableaux. We can thus identify the particular step in the sequence at which each new tableau is created, and we can enumerate the tableaux \( \mathcal{T}_0, \mathcal{T}_1, \mathcal{T}_2, \ldots \), in this order. In addition, for each tableau \( \mathcal{T}_n \), we can identify the immediate predecessor of \( \mathcal{T}_n \) and the immediate successors of \( \mathcal{T}_n \) under the operation of the rules \( P \Leftarrow \neg Q \) and \( P \Leftarrow [Q \Rightarrow R] \). Obviously, the immediate prede-
cessor of $\mathcal{I}_n$ must be created before $\mathcal{I}_n$ can be created, and must therefore also precede $\mathcal{I}_n$ in the enumeration.

Our first step is to construct a set of substates $S(n) = \{ s(n, k) \mid k < \omega \}$ for each of these tableaux $\mathcal{I}_n$. The set $S(0)$ associated with the initial tableau $\mathcal{I}_0$ will be a singleton set containing the substate $s(0, 0)$, but the subsequent sets $S(n)$ will be infinite in general. To construct the set $S(n)$, for $n > 0$, it is necessary to assign ground terms in $U$ to all variables $?x$ and $!y$ that appear in the tableaux $\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_n$. For existential variables in the form $!y$ this is easy: There is at most a denumerable set of these variables, so we pick a denumerable set of individuals $\{ c_i \}$ that do not appear in the query $P(x)$, or in any ground instance in $s_0$, or in any rules in $\mathcal{R}$, and we let $\psi$ be any one-to-one mapping from $\{ !y_i \}$ to $\{ c_i \}$. Then, when we need a substitution for some finite set of existential variables $\{ !y_i \}$, we can simply use the mapping $\psi$ restricted to the finite set. For local tableau variables in the form $?x$ the task is more difficult. We will proceed here, by induction, in the order in which the tableaux are created, and define a set of \textit{data-base substitutions} $\Theta(n) = \{ \theta(n, k) \mid k < \omega \}$ to go along with the set of substates $S(n)$ associated with each tableau $\mathcal{I}_n$.

Since there are no variables in the form $?x$ in the initial tableau, the data-base substitution $\theta(0, 0)$ associated with the substate $s(0, 0)$ in $S(0)$ will be the identity mapping. Assume now that the set of substates $S(m)$ has been defined for the tableau $\mathcal{I}_m$ that is an immediate predecessor of $\mathcal{I}_n$, and assume that a data-base substitution $\theta(m, j)$ has been associated with each $s(m, j)$ in $S(m)$. Let $\sigma_n$ be any answer substitution that is returned from $\mathcal{I}_n$ to its immediate predecessor $\mathcal{I}_m$ at any point in the sequence of rule applications as the result of the closure of a principal refutation tree beginning with $F$. If $\sigma_n$ itself contains existential variables in the form ‘$!y_i$’, we will form the composition of $\sigma_n$ with the mapping $\psi$ restricted to the finite set of variables $\{ !y_i \}$, and we will let $\Sigma_n$ be the set of all such composite substitutions $\sigma_n \circ \psi$. Now pick any $s(m, j)$ and its associated $\theta(m, j)$. Let $\theta_n$ be any ground substitution for the new variables in the form $?x$ that are introduced in the tableau $\mathcal{I}_n$, but restricted to the terms that can be constructed from the constants in $s_0$ and $\mathcal{R}$, plus the constants $c_i = \psi(\{ !y_i \})$ for the existential variables ‘$!y_i$’ that were introduced in $\mathcal{I}_m$ and its predecessors. Consider the composition of $\theta(m, j)$ with $\theta_n$. If $\theta(m, j) \circ \theta_n$ is consistent with one of the substitutions $\sigma_n \circ \psi$ in $\Sigma_n$, then we will do nothing. On the other hand, if $\theta(m, j) \circ \theta_n$ is not consistent with any of the substitutions $\sigma_n \circ \psi$ in $\Sigma_n$, then we will construct a new substate $s'$ as one of the \textit{immediate successors} of $s(m, j)$ in $S(n)$, and we will let $\theta(m, j) \circ \theta_n$ be the data-base substitution associated with $s'$. Clearly, if we can do this for each $s(m, j)$ and its associated $\theta(m, j)$, and for each $\theta_n$, then we can define the desired sets $S(n)$ and $\Theta(n)$ by a simultaneous induction. We note, too, that $S(n)$ and $\Theta(n)$ will always be denumerable sets.

It remains to be seen, though, how the content of each substate in $S(n)$ is to be defined. Let us use the expression $s'(m, j, \theta_n)$ to denote the immediate successor of $s(m, j)$ for a particular $\theta_n$. We want $s'(m, j, \theta_n)$ to be the smallest substate containing $s(m, j)$ that satisfies the following three conditions:

(1) Intuitively, we want $s'(m, j, \theta_n)$ to satisfy the rules $P \leftarrow \bigwedge Q_j$ and $P \leftarrow \bigvee Q_j$. This is straightforward, however, and we simply stipulate that this should be the case.
(2) Intuitively, we want \( s'(m, j, \theta_n) \) to include the ground instantiations of the data-base formulae in \( T_n \) and its predecessors, except for those ground instantiations that have been returned as answer substitutions to the immediate predecessor \( T_m \) as the result of the closure of a principal refutation tree beginning with \( F \). But the data-base substitution \( \theta(m, j) \circ \theta_n \) has been defined with just this purpose in mind. Accordingly, let \( Q(?x; !y) \) be the formula in the data base of \( T_n \), and let \( \psi \) be our existential mapping restricted to the variables \('!y'\) in \(!y\). Then we require that

\[
Q(?x;!y) \circ \theta(m, j) \circ \theta_n \subseteq s'(m, j, \theta_n).
\]

(3) Finally, we want \( s'(m, j, \theta_n) \) to include all ground instances that are returned from the immediate successors of \( T_n \) because of the satisfaction of the rules \( P \Leftarrow \neg Q \) and \( P \Leftarrow [Q \Rightarrow R] \). Formally we do this as follows: Let \( \sigma_a \) be any answer substitution returned to \( T_n \) at any point in the sequence of rule applications as a result of the closure of an auxiliary tableau that was generated by a negation or embedded implication rule with the atomic formula \( P(x) \) on its left-hand side. Although the prefix '?' will have been stripped away from all variables ?x in \( \sigma_a \) that were created in the auxiliary tableau itself, there may be some ?x variables and !y variables remaining in \( \sigma_a \) that were created in \( T_n \) and its predecessors. Let \( \psi \) be our existential mapping restricted to the variables \('!y'\) in \(!y\). If \( \theta(m, j) \circ \theta_n \) is inconsistent with the bindings for ?x in \( \sigma_a \circ \psi \), then we will do nothing. Otherwise, if \( \theta(m, j) \circ \theta_n \) is consistent with the bindings for ?x in \( \sigma_a \circ \psi \), then form

\[
P(x) \sigma_a \circ \psi \circ \theta(m, j) \circ \theta_n =
\]

and interpret this as a Horn clause with a null antecedent that \( s'(m, j, \theta_n) \) must satisfy.

Although these three conditions are formulated for tableaux \( T_n \) with \( n > 0 \), they reduce to a particularly simple form for the initial tableau \( T_0 \). We have already stipulated that \( S(0) \) is the singleton set \{ \( s(0, 0) \} \). The first condition simply requires that \( s(0, 0) \) satisfies the rules \( P \Leftarrow \forall Q_j \) and \( P \Leftarrow \forall Q_j \), and the second condition requires that \( s(0, 0) \) contain the initial substate \( s_0 \). The third condition is applicable as written, except that the substitution \( \sigma_a \) returned to the tableau \( T_0 \) will contain no variables in the form ?x or !y, and so the Horn clause with null antecedent reduces to \( P(x) \sigma_a \circ \psi \). Since this is exactly the result that we want for \( S(0) \), we can use these three conditions equally well for both the basis step and the inductive step of our definition of \( S(n) \).

It follows, then, that \( S(n) \) is well defined if each \( s'(m, j, \theta_n) \) is well defined. Is there a unique minimal substate \( s'(m, j, \theta_n) \geq s(m, j) \) that satisfies these three conditions? Note that the set of rules constraining \( s'(m, j, \theta_n) \) by the third condition could very well be infinite. But all these rules are Horn clauses, and we know that Theorem 2.9 holds even if the initial substate \( s_0 \) and the set of rules \( R \) are infinite. Thus, by Theorem 2.9, there exists a unique minimal substate \( s' \geq s(m, j) \) that satisfies all three conditions. It follows that our construction \( S(n) = \{s(n, k) | k < \omega \} \) is well defined for every \( n \).

We will now establish a completeness lemma for principal refutation trees, using a similar idea. Since each \( s(n, k) \) is a unique minimal substate generated by a
(possibly infinite) set of Horn clauses, we can apply Theorem 2.14 (completeness) to construct a finite closed refutation tree for any ground instance in \( s(n, k) \). This refutation tree can then be used to construct a principal refutation tree in the tableau \( \mathcal{T}_n \). The completeness lemma for principal refutation trees is stated as follows:

Lemma 4.8. Assume that \( P_0(?x; !y; z_0) \psi \circ \theta(n, k) \circ \theta \in s(n, k) \) for some ground substitution \( \theta \). Then any refutation tree in \( \mathcal{T}_n \) beginning with \( P_0(?x; !y; z_0) \) will eventually close with an answer substitution \( \sigma(z_0) \) for the variables \( z_0 \) such that \( \sigma(z_0) \circ \psi \circ \theta(n, k) \leq \theta \), and an answer substitution \( \sigma(\alpha) \) for the local tableau variables in \( \mathcal{T}_n \) and its predecessors such that \( \sigma(?x) \circ \psi \leq \theta(n, k) \).

Proof: Assume that \( P_0(?x; !y; z_0) \psi \circ \theta(n, k) \circ \theta \in s(n, k) \) for some ground substitution \( \theta \). Since \( P_0(?x; !y; z_0) \psi \circ \theta(n, k) \) is a formula depending only on \( z_0 \), we know from Theorem 2.14 (completeness) that there exists a finite closed refutation tree for \( P_0(?x; !y; z_0) \psi \circ \theta(n, k) \) with an answer substitution \( \sigma' \) such that \( \sigma'(z_0) \leq \theta \). Let us call this refutation tree \( T' \). The branches of \( T' \) would be constructed from rules in the form \( P \leftarrow \land Q \) and \( P \leftarrow \lor Q \), and each branch would terminate in either: (1) a ground instance in \( s_0 \), or a Horn clause \( P(x) \leftarrow \) with a null antecedent; (2) a ground instance \( Q(?x; !y) \psi \circ \theta(n, k) \) that was added to \( s(n, k) \) by the second condition of our construction; or (3) a rule \( P(x) \alpha \circ \psi \circ \theta(n, k) \leftarrow \) with a null antecedent that was added by the third condition of our construction. [Note: It is possible that the particular rule \( P(x) \alpha \circ \psi \circ \theta(n, k) \) was added in some predecessor of \( s(n, k) \), but in this case there would exist an identical rule, with renamed variables, that would eventually be added to \( s(n, k) \). We may thus assume, without loss of generality, that the rule \( P(x) \alpha \circ \psi \circ \theta(n, k) \) was added to \( s(n, k) \) because of the closure of an auxiliary tableau attached directly to \( \mathcal{T}_n \).] To prove the lemma, we will show that we can systematically “invert” the substitutions \( \theta(n, k) \) and \( \psi \), and replace the ground terms in \( T' \) with the original variables \( ?x \) and \( !y \). We will obtain in this fashion a new refutation tree, call it \( T \), which is composed entirely of the rules and the data from \( \mathcal{T}_n \) and its predecessors, and which would thus be constructed by any fair sequence of rule applications beginning with the node \( P_0(?x; !y; z_0) \).

Consider any arbitrary branch of \( T' \). At the top of this branch, the node containing \( P_0(?x; !y; z_0) \psi \circ \theta(n, k) \) must have unified with some formula \( P_1(z_1) \) on the left-hand side of one of the rules \( P \leftarrow \land Q \) or \( P \leftarrow \lor Q \). Thus, if \( \alpha_1 \) is the unifying substitution, we know that

\[
P_0(?x; !y; z_0) \psi \circ \theta(n, k) \circ \alpha_1 = P_1(z_1) \alpha_1.
\]

Since \( \theta(n, k) \) includes none of the variables \( z_1 \), we can rewrite this equality as

\[
P_0(?x; !y; z_0) \psi \circ \theta(n, k) \circ \alpha_1 = P_1(z_1) \theta(n, k) \circ \alpha_1,
\]

from which it is apparent that \( P_0(?x; !y; z_0) \psi \) and \( P_1(z_1) \) are unifiable with a most general unifier \( \sigma_1 \leq \theta(n, k) \circ \alpha_1 \). This means that there must exist some substitution \( \rho_1 \) such that

\[
\sigma_1 \circ \rho_1 = \theta(n, k) \circ \alpha_1.
\]

We have thus “inverted” the top node of the refutation tree \( T' \).
Now assume that the selected branch of $T'$ was constructed from a chain of rules $P \leftarrow \land Q_j$ or $P \leftarrow \lor Q_j$, so that each node contains a formula $Q_{j-1}(z_{j-1})\sigma_{j-1}$ that unifies with some formula $P_j(z_j)$ from the left-hand side of one of the rules. We wish to "invert" each of these nodes, and compute for each formula $Q_{j-1}(z_{j-1})\sigma_{j-1}$ and each formula $P_j(z_j)$ a most general unifier $\sigma_j$, which might very well include $?x$ variables. We thus need to prove that there exists a sequence of most general unifiers $\sigma_1, \sigma_2, \ldots, \sigma_j$ along the nodes of our selected branch, such that

$$\sigma_1 \circ \rho_1 = \theta(n, k) \circ \sigma'_1,$$

$$\sigma_2 \circ \rho_2 = \rho_1 \circ \sigma'_2,$$

$$\vdots$$

$$\sigma_j \circ \rho_j = \rho_{j-1} \circ \sigma'_j$$

for some substitutions $\rho_1, \rho_2, \ldots, \rho_j$. The proof is by induction. The base case has been established in (9) above. So assume that $Q_{j-2}(z_{j-2})\sigma_{j-2}$ and $P_{j-1}(z_{j-1})$ are unifiable with a most general unifier $\sigma_{j-1}$, and assume that

$$\sigma_{j-1} \circ \rho_{j-1} = \rho_{j-2} \circ \sigma'_{j-1}$$

for some substitution $\rho_{j-1}$. Let $Q_{j-1}(z_{j-1})\sigma'_{j-1}$ be the formula in the next node of the selected branch in $T'$, and assume that $Q_{j-1}(z_{j-1})\sigma'_{j-1}$ unifies with some formula $P_j(z_j)$ from the left-hand side of one of the rules $P \leftarrow \land Q_j$ or $P \leftarrow \lor Q_j$. Let $\sigma'_j$ be the unifying substitution at this node in the selected branch of $T'$. Then

$$Q_{j-1}(z_{j-1})\sigma'_{j-1} \circ \sigma'_j = P_j(z_j)\sigma'_j,$$

which can be rewritten as

$$Q_{j-1}(z_{j-1})\rho_{j-2} \circ \sigma'_{j-1} \circ \sigma'_j = P_j(z_j)\rho_{j-1} \circ \sigma'_j,$$

since $\rho_{j-2}$ includes none of the variables $z_{j-1}$, and since $\rho_{j-1}$ includes none of the variables $z_j$. Using Equation (11), this equation becomes

$$Q_{j-1}(z_{j-1})\sigma_{j-1} \circ \rho_{j-1} \circ \sigma'_j = P_j(z_j)\rho_{j-1} \circ \sigma'_j,$$

from which it is apparent that $Q_{j-1}(z_{j-1})\sigma_{j-1}$ and $P_j(z_j)$ are unifiable with a most general unifier $\sigma_j \leq \rho_{j-1} \circ \sigma'_j$. In other words,

$$\sigma \circ \rho_j = \rho_{j-1} \circ \sigma'_j$$

for some substitution $\rho_j$. This completes the inductive proof of the equations in (10).

There are now three cases to consider. If the selected branch of $T'$ terminates in a ground instance in $s_0$, or if it terminates because one of the rules $P \leftarrow \land Q_j$ or $P \leftarrow \lor Q_j$ has a null antecedent, then our analysis of the branch is essentially complete. In this case, the answer substitution along the selected branch in $T'$ is simply

$$\sigma' = \sigma'_j \circ \sigma'_2 \circ \cdots \circ \sigma'_1,$$

and by forming the composition of the most general unifiers in (10), we can see that

$$\sigma_1 \circ \sigma_2 \circ \cdots \circ \sigma_j \circ \rho_j = \theta(n, k) \circ \sigma'_1 \circ \sigma'_2 \circ \cdots \circ \sigma'_j.$$  

Let us now set

$$\sigma = \sigma_1 \circ \sigma_2 \circ \cdots \circ \sigma_j \circ \psi^{-1},$$
where \( \psi^{-1} \) simply inverts the substitution \( \psi \) and replaces the constants \( \{ c_i \} \) in each unifier \( \sigma \) with the original variables \( !y \). From an inspection of Equation (12), we can easily verify that

\[
\sigma(z_0) \circ \psi \circ \theta(n, k) \leq \sigma'
\]

and

\[
\sigma(?x) \circ \psi \leq \theta(n, k).
\]

Thus \( \sigma \) is the desired answer substitution along the selected branch of the inverted refutation tree \( T \).

Next assume that the selected branch of \( T' \) terminates when the formula \( Q_j(z_j) \sigma'_j \) unifies with a ground instance \( Q(\bar{x}; !y) \psi \circ \theta(n, k) \) that was added by the second condition in our construction of \( s(n, k) \). If the unifying substitution is \( \sigma'_{j+1} \), we know that

\[
Q_j(z_j) \sigma'_j \circ \sigma'_{j+1} = Q(\bar{x}; !y) \psi \circ \theta(n, k) \circ \sigma'_{j+1},
\]

which can then be rewritten as

\[
Q_j(z_j) \sigma_j \circ \rho_j \circ \sigma'_{j+1} = Q(\bar{x}; !y) \psi \circ \theta(n, k) \circ \sigma'_{j+1}
\]

by the same arguments used above. From an induction on the recurrence relations in (10), it is easy to verify the following proposition: Any bindings for the \( ?x \) variables that are included in \( \rho_j \) must be identical to the bindings in \( \theta(n, k) \). Furthermore, if \( \sigma \) includes any bindings for the variables \( z_j \) that contain variables in the form \( ?x \), then \( \rho_j \) must also include bindings for these \( ?x \) variables. As a result, we can rewrite our last equation as

\[
Q_j(z_j) \sigma_j \circ \rho_j \circ \theta(n, k) \circ \sigma'_{j+1} = Q(?x; !y) \psi \circ \theta(n, k) \circ \sigma'_{j+1},
\]

from which it is apparent that \( Q_j(z_j) \sigma_j \) and \( Q(?x; !y) \psi \) have a most general unifier \( \sigma_{j+1} \) such that

\[
\sigma_{j+1} \circ \rho_{j+1} = \rho_j \circ \theta(n, k) \circ \sigma'_{j+1}
\]

for some substitution \( \rho_{j+1} \). We can now extend Equation (12) to

\[
\sigma_1 \circ \sigma_2 \circ \cdots \circ \sigma_j \circ \sigma_{j+1} \circ \rho_{j+1} = \theta(n, k) \circ \sigma'_1 \circ \sigma'_2 \circ \cdots \circ \sigma'_j \circ \sigma'_{j+1},
\]

and set \( \sigma = \sigma_1 \circ \sigma_2 \circ \cdots \circ \sigma_j \circ \sigma_{j+1} \circ \psi^{-1} \) as before. Once again, we can see that \( \sigma \) is the desired answer substitution along the selected branch of the inverted refutation tree \( T \), and that it has the desired properties.

Finally, consider a branch of \( T' \) that terminates by unification with an atomic formula \( P(z_{j+1}) \sigma_a \circ \psi \circ \theta(n, k) \) that was added by the third condition of our construction. Since the predicate on the left-hand side of every negation or embedded implication is unique, the formula under consideration must have been derived from a particular rule \( P \Rightarrow \neg Q \) or \( P \Rightarrow [Q \Rightarrow R] \), and it must have resulted from the closure of a particular auxiliary tableau with answer substitution \( \sigma_a \). By virtue of our construction, \( \sigma_a \) contains no variables from \( z_0, z_1, \ldots, z_j \), and the bindings for the \( ?x \) variables included in \( \sigma_a \circ \psi \) must be consistent with \( \theta(n, k) \). Thus, by all of our previous arguments, if \( \sigma'_{j+1} \) is the most general unifier of \( Q_j(z_j) \sigma'_j \) and \( P(z_{j+1}) \sigma_a \circ \psi \circ \theta(n, k) \) in the selected branch of \( T' \), then the following equation must hold:

\[
Q_j(z_j) \sigma_j \circ \rho_j \circ \theta(n, k) \circ \sigma'_{j+1} = P(z_{j+1}) \sigma_a \circ \psi \circ \rho_j \circ \theta(n, k) \circ \sigma'_{j+1}.
\]
It then follows, as before, that \( Q_j(z_j)\sigma_j \) and \( P(z_{j+1})\sigma_a \circ \psi \) have a most general unifier \( \sigma_{j+1} \) such that

\[
\sigma_{j+1} \circ \rho_{j+1} = \rho_j \circ \theta(n, k) \circ \sigma_{j+1}
\]

for some substitution \( \rho_{j+1} \). In this case, however, in order to conclude that the selected branch will terminate in the inverted tree \( T \), we must also verify that \( Q_j(z_j)\sigma_j \) and \( P(z_{j+1})\sigma_a \circ \psi \) have a unifying composition with respect to the variables \( ?x \). But since \( \sigma_j(?x) \leq \theta(n, k) \) by the recurrence relations (10), and since \( \sigma_a \circ \psi \) is consistent with \( \theta(n, k) \) by construction, it is easy to see that this condition is satisfied.

Let us now summarize these results. For each branch in the original refutation tree \( T' \) that terminates with an answer substitution \( \sigma' \), we have constructed a corresponding branch in the inverted refutation tree \( T \) that terminates with an answer substitution \( \sigma \), and we have shown that

\[
\sigma(z_0) \circ \psi \circ \theta(n, k) \leq \sigma'
\]

and

\[
\sigma(?x) \circ \psi \leq \theta(n, k)
\]

It is now easy to see that there exists a unifying composition of these answer substitutions across all the branches of \( T \). For we know that there exists a unifying composition of the answer substitutions \( \sigma' \) for the variables \( z_0 \) across all the branches of \( T' \), and the relationships above show that this unifying composition will still go through when we "invert" the substitutions \( \theta(n, k) \) and \( \psi \). We have thus shown that there exists a refutation tree \( T \) for \( P_0(?x; !y; z_0) \), with a grand answer substitution \( \sigma \) at the top, such that

\[
\sigma(z_0) \circ \psi \circ \theta(n, k) \leq \theta
\]

and

\[
\sigma(?x) \circ \psi \leq \theta(n, k).
\]

Now assume, at some point in our sequence of rule applications, that we have initiated the construction of a new refutation tree in the tableau \( \mathcal{T}_n \) by writing the atomic formula \( P_0(?x; !y; z_0) \) in its topmost node. Since each component of \( T \) can be constructed by following the basic tableau extension and closure steps, and since the sequence of rule applications is assumed to be fair, it follows that our proof procedure will eventually construct the entire refutation tree \( T \) in the tableau \( \mathcal{T}_n \) and place the answer substitution \( \sigma \) at its top. This completes the proof of Lemma 4.8. \( \square \)

At this point, we have constructed a set of substates \( S(n) = \{ s(n, k) \mid k < \omega \} \) for each tableau \( \mathcal{T}_n \), and we have shown that these substates satisfy the completeness lemma for principal refutation trees. Let us now construct the set

\[
K = \bigcup \{ S(n) \mid n < \omega \}.
\]

We note that \( K \) is partially ordered by set inclusion, and that \( s(m, j) \leq s'(m, j, \theta_n) \). We claim that \( K \) has the following additional properties: \( K \subseteq J_0 \) and \( K \subseteq T_N(K) \). We
will establish these properties in the next two lemmas:

**Lemma 4.9.** \( K \subseteq J_0 \).

**Proof.** We will use the definition of \( J_0 \) given in the proof of Theorem 3.4. It is obvious that \( K \subseteq G \). We therefore have to show that every rule \( P \Rightarrow \neg Q \) is satisfied in every substate \( s(n, k) \in K \). We will do the proof here for \( n > 0 \), since the proof for the substate \( s(0, 0) \) then follows in a similar manner as a special case.

Let us select a particular rule \( P(z_0) \Rightarrow \neg Q(z_0; z_1) \) for analysis, and let us consider any substate \( s(n, k) \in S(n) \) such that \( P(z_0) \theta_0 \in s(n, k) \) for some ground substitution \( \theta_0 \). We will analyze the two open nodes of the principal refutation tree generated by the rule \( F \equiv P(z_0) \land Q(z_0; z_1) \) in the tableau \( T_n \). By Lemma 4.8, the node beginning with \( P(z_0) \) will eventually close with an answer substitution \( \sigma_0 \) such that

\[
\sigma_0(z_0) \circ \psi \circ \theta(n, k) \leq \theta_0
\]

and

\[
\sigma_0(\exists x) \circ \psi \leq \theta(n, k).
\]

Now assume that there exists a ground substitution \( \theta_1 \geq \theta_0 \) such that \( Q(z_0; z_1) \theta_1 \in s(n, k) \). By Lemma 4.8 again, the node beginning with \( Q(z_0; z_1) \) will eventually close with an answer substitution \( \sigma_1 \) such that

\[
\sigma_1(z_0; z_1) \circ \psi \circ \theta(n, k) \leq \theta_1
\]

and

\[
\sigma_1(\exists x) \circ \psi \leq \theta(n, k).
\]

But it is now easy to verify that \( \sigma_0 \) and \( \sigma_1 \) have a unifying composition \( \sigma \) such that \( \sigma(\exists x) \circ \psi \leq \theta(n, k) \). Thus the tableau \( T_n \) will eventually close and return the answer substitution \( \sigma \) to its predecessor, assuming here that \( n > 0 \). However, this contradicts our initial choice of \( \theta(n, k) \) as a ground substitution that is inconsistent with \( \sigma \circ \psi \), for every answer substitution \( \sigma \) returned from \( T_n \) as the result of the closure of a principal refutation tree beginning with \( F \). We conclude from this contradiction that \( Q(z_0; z_1) \theta_1 \in s(n, k) \) for every \( \theta_1 \geq \theta_0 \). For the case \( n = 0 \), of course, we can derive a similar contradiction from our initial assumption that the tableau \( T_0 \) never closes at all.

We have thus shown that the substate \( s(n, k) \) by itself satisfies all rules in the form \( P \Rightarrow \neg Q \). For substates \( s' \supseteq s(n, k) \) we can extend these arguments as follows: We note that \( P(z_0) \theta_0 \in s' \) whenever \( P(z_0) \theta_0 \in s(n, k) \), and that the principal refutation tree that we constructed in \( T_n \) would also be constructed in the tableau associated with \( s' \). Thus the arguments above apply equally well to all substates \( s' \supseteq s(n, k) \). This completes the proof of Lemma 4.9. \( \square \)

**Lemma 4.10.** \( K \subseteq T_N(K) \).

**Proof.** We will use Definition 3.5, and show that every substate \( s \in K \) is preserved by the application of \( T_N \) to \( K \). Without loss of generality, we may assume that the substate \( s \) is a particular substate \( s(m, j) \in S(m) \), which was defined in connection
with the tableau \( \mathcal{T}_m \), and which has a data-base substitution \( \theta(m, j) \) associated with it. We can thus analyze the substates \( s'(m, j, \theta_n) \) that were constructed as immediate successors to \( s(m, j) \), and then renamed \( s(n, k) \).

Consider an arbitrary negation rule \( P(x) \Leftarrow \neg Q(x; y) \) that is extended at some point in the sequence of rule applications from the tableau \( \mathcal{T}_m \) to a new auxiliary tableau \( \mathcal{T}_n \). We will apply the first half of Definition 3.5 to the substate \( s(m, j) \), and attempt to show that this definition holds for the selected negation rule. Let \( \theta \) be any ground substitution for the variables in \( P(x) \) restricted to the terms in \( U(s(m, j)) \). From our construction, it is easy to see that the substate \( s(m, j) \) contains no individual constants in the set \( \{ c_i \} \) unless \( c_i = \psi(\!y_i) \) for some existential variable \( \!y_i \) that was introduced in \( \mathcal{T}_m \) or one of its predecessors. Thus the substitutions \( \theta_n \) that we used in the construction of the substates \( s' \geq s(m, j) \) would include all possible values of \( \theta \). There are two cases to consider here: If \( \theta(m, j) \circ \theta_n \) is inconsistent with \( \sigma_n \circ \psi \) for every answer substitution \( \sigma_n \) returned from \( \mathcal{T}_n \) as the result of the closure of a principal refutation tree beginning with \( \mathbf{F} \), then we have constructed a new substate \( s'(m, j, \theta_n) \geq s(m, j) \) in \( \mathcal{K} \), and we have stipulated that

\[
Q(?x; \!y) \psi \circ \theta(m, j) \circ \theta_n \in s'(m, j, \theta_n)
\]

by the second condition of our construction. On the other hand, if \( \theta(m, j) \circ \theta_n \) is consistent with \( \sigma_n \circ \psi \) for some answer substitution \( \sigma_n \) returned from the closure of a principal refutation tree in \( \mathcal{T}_n \) beginning with \( \mathbf{F} \), then \( \theta(m, j) \) is also consistent with \( \sigma_n \circ \psi \), and we have stipulated that the substate \( s(m, j) \) satisfies the rule

\[
P(x) \sigma_n \circ \psi \circ \theta(m, j) \Leftarrow
\]

by the third condition of our construction, in the prior step of our inductive definition. In this case, it follows that \( P(x) \theta \in s(m, j) \). In either case, we can see that the first half of Definition 3.5 holds.

Now consider an arbitrary embedded implication rule \( P(x) \Leftarrow [Q(x; y) \Rightarrow R(x; y)] \) that is extended at some point in the sequence of rule applications from the tableau \( \mathcal{T}_m \) to a new auxiliary tableau \( \mathcal{T}_n \), and apply the second half of Definition 3.5 to the substate \( s(m, j) \). Let \( \theta \) again be any ground substitution for the variables in \( P(x) \) restricted to the terms in \( U(s(m, j)) \), and consider the corresponding substitution \( \theta_n \) in our construction of the substates \( s' \geq s(m, j) \). If \( \theta(m, j) \circ \theta_n \) is consistent with \( \sigma_n \circ \psi \) for some answer substitution \( \sigma_n \) that resulted from the closure of a principal refutation tree in \( \mathcal{T}_n \) beginning with \( \mathbf{F} \), then the analysis proceeds exactly as in the case of the negation rules. Otherwise, there will exist a substate \( s'(m, j, \theta_n) \geq s(m, j) \) in \( \mathcal{K} \) such that

\[
Q(?x; \!y) \psi \circ \theta(m, j) \circ \theta_n \in s'(m, j, \theta_n).
\]

Now assume that the second condition of Definition 3.5 is satisfied. It then follows that

\[
R(?x; \!y) \psi \circ \theta(m, j) \circ \theta_n \in s'(m, j, \theta_n),
\]

and we can apply Lemma 4.8 to the principal refutation tree in \( \mathcal{T}_n \) beginning with \( R(?x; \!y) \). According to Lemma 4.8, this principal refutation tree will eventually close with an answer substitution \( \sigma \) such that \( \sigma(?x) \circ \psi \leq \theta(m, j) \circ \theta_n \), and when the proof procedure strips away the prefix "?" from the variables \(?x\) that were created in \( \mathcal{T}_n \), it returns an answer substitution \( \sigma_n \) from \( \mathcal{T}_n \) such that \( \sigma_n \circ \psi \) is consistent with
Thus, by the third condition of our construction, the substate \( s(m, j) \) must satisfy the rule

\[
P(x) \sigma_n \circ \psi \circ \theta(m, j) \Rightarrow ,
\]

and we can see again that the second half of Definition 3.5 holds.

Since these arguments apply to any arbitrary rules \( P \Leftarrow Q \) and \( P \Leftarrow \{Q \Rightarrow R\} \), it follows that the substate \( s(m, j) \) is preserved by the application of \( T_N \) to \( K \). This completes the proof of Lemma 4.10. \( \square \)

Let us now recall the definition of \( K^* \) in Theorem 3.7:

\[
K^* = \bigcup \{ J | J \subseteq T_N(J) \cap J_0 \}.
\]

It is clear from Lemma 4.9 and Lemma 4.10 that \( K \subseteq K^* \). In particular, the substate \( s(0, 0) \) associated with the initial tableau \( T_0 \) must be a member of \( K^* \). Suppose \( P(x) \theta \in s(0, 0) \). Note that the data base of \( T_0 \) includes only the ground instances in \( s_0 \), and note that the answer substitutions \( \sigma_n \) returned to \( T_0 \) contain no variables in the form \( ?x \) or \( !y \). Thus, by Lemma 4.8, if \( P(x) \theta \in s(0, 0) \), then the principal refutation tree in \( T_0 \) beginning with \( P(x) \) will eventually close with an answer substitution \( \sigma \) such that \( \sigma(x) \leq \theta \). But this contradicts the initial assumption in the proof that our chosen sequence of rule applications never closes. We thus conclude that \( P(x) \theta \notin s(0, 0) \). This completes the proof of Theorem 4.6.

We will now state and prove a simple corollary. Theorem 4.6 asserts the existence of a closed system of tableaux, but we can use the same techniques to show that every attempt to construct a system of tableaux for a formula that is uniformly entailed eventually closes, as long as the sequence of rule applications is fair.

**Corollary 4.11 (Strong completeness).** Let \( s_0 \) be a finite initial substate in \( B \), and let \( \mathcal{R} \) be a finite set of rules including negations and embedded implications. Assume that \( (3x)P(x) \) is uniformly entailed by \( s_0 \) and \( \mathcal{R} \) with ground substitution \( \theta \). Then every fair sequence of rule applications starting with the initial tableau for \( P(x) \) will eventually close with an answer substitution \( \sigma \) such that \( \sigma(x) \leq \theta \).

**Proof.** Pick any fair sequence of rule applications, and assume that the systems of tableaux generated by these rules never close. Then the construction of a substate \( s \in K^* \) such that \( P(x) \theta \notin s \) follows exactly as in the proof of Theorem 4.6. \( \square \)

### 4.5. Fixed Points Revisited

In this subsection, we will return to the fixed-point semantics of Section 3 (Paper I), and show that the conclusion of Theorem 3.15 can be sharpened under certain conditions.

We first establish two corollaries to the soundness and completeness theorems. Corollary 4.12 is a generalization of Corollary 2.13 in the pure Horn-clause case, and Corollary 4.13 is a sharpened version of the inequality in Theorem 3.15. In both of these corollaries, we will assume, as we did in Theorem 4.6, that the initial substate \( s_0 \) and the rules \( \mathcal{R} \) are finite. However, we will subsequently remove these restrictions.
Corollary 4.12. If $K^*$ is nonempty and $A \in \cap K^*$, then all the individual constants in $A$ appear somewhere in the initial substate $s_0$, or in the rules $\mathcal{R}$.

PROOF. To prove the corollary, we simply repeat the construction in Theorem 4.6, starting with an initial tableau $\mathcal{T}_0$ for the query $A$. In this case, though, we construct an infinite sequence of tableaux $\mathcal{T}_0, \mathcal{T}_1, \mathcal{T}_2, \ldots$, regardless of closure, we define the set $K = \cup \{ S(n) \mid n < \omega \}$ exactly as before, and we then analyze the substate $s(0,0)$ associated with the tableau $\mathcal{T}_0$. Recall that the data base of $\mathcal{T}_0$ includes only the ground instances in $s_0$, and recall that the answer substitutions $\sigma_a$ returned to $\mathcal{T}_0$ contain no variables in the form $?x$ or $!y$. Thus the Horn clauses that are used to construct the substate $s(0,0)$ contain, at most, the individual constants from the rules in $\mathcal{R}$. By Corollary 2.13 to Theorem 2.12, this means that all the individual constants in all the ground atomic formulae in $s(0,0)$ appear somewhere in $s_0$ or $\mathcal{R}$.

Now, since $K^*$ is nonempty, we know from Lemma 4.4 (soundness) that the principal refutation trees in $\mathcal{T}_0$ beginning with $F$ never close. We can thus apply Lemma 4.9 and Lemma 4.10, exactly as in the proof of Theorem 4.6, and conclude that $s(0,0) \in K^*$. So if there is an individual constant in $A$ that does not appear in $s_0$ or $\mathcal{R}$, then $A \notin s(0,0)$, and thus $A \notin \cap K^*$, which is a contradiction. □

To prove a sharpened version of the inequality in Theorem 3.15, we will use both the soundness theorem and the completeness theorem. However, the most general form of Lemma 4.4 (soundness) includes an assumption about the individual constants in $\mathcal{R}$, and we must adopt this assumption here in order to get the following proof to work.

Corollary 4.13. Let $s_0$ be a finite substate in $B$, and let $\mathcal{R}$ be a finite set of rules including negations and embedded implications. Assume that the individual constants in $\mathcal{R}$ appear in $s_0$. Then the inequality in Theorem 3.15 becomes an equality:

$$\cap K^* = \cap J_\omega = \sqcup \{ \cap T^k_N(J_0) \mid k < \omega \}.$$

PROOF. In Theorem 3.15, under the assumption that $K^*$ is nonempty, we showed that $\cap K^* \geq \cap J_\omega \geq \sqcup \{ \cap T^k_N(J_0) \mid k < \omega \}$. Thus we need to establish the opposite inequality here.

Assume that $A$ is a ground instance in $\cap K^*$. By Theorem 4.6 (completeness), there exists a closed system of tableaux for $A$ with some answer substitution $\sigma$. Assume that the substitution $\sigma$ has degree $m$, and apply Lemma 4.4 (soundness). According to this lemma, $A \sigma \in s$ for every substate $s \in T^m_N(J_0)$ that includes the individual constants in $A$ and $\mathcal{R}$. However, by Corollary 4.12, all the individual constants in $A$ appear in $s_0$ or $\mathcal{R}$, and by assumption all the individual constants in $\mathcal{R}$ appear in $s_0$. Thus every substate $s \in T^m_N(J_0)$ includes the individual constants in $A$ and $\mathcal{R}$, and it follows that $A \in \cap T^m_N(J_0)$. We have thus shown that

$$\cap K^* \leq \sqcup \{ \cap T^k_N(J_0) \mid k < \omega \}$$

as required. □

It is an open question whether or not the equality holds without imposing this restriction on the individual constants in $\mathcal{R}$.
We will now reconsider the restriction of Corollary 4.13 to a finite \(s_0\) and a finite \(\mathcal{R}\). In the completeness theorem itself, this was not a significant restriction, since we would never use an infinite set of rules in a practical proof procedure. However, for an analysis of the inequality in Theorem 3.15, and for other theoretical purposes, it is important to know what happens when \(s_0\) and \(\mathcal{R}\) become infinite. We know that Corollary 4.13 cannot be true for arbitrary \(s_0\) and \(\mathcal{R}\), since we have seen a counterexample in Section 3.4. The question is: What is the weakest condition on \(s_0\) and \(\mathcal{R}\) that still preserves the equality in Corollary 4.13?

Examining the proof of Theorem 4.6, we see that the finiteness assumption enters the analysis at two points: First, we used the finiteness of \(s_0\) and \(\mathcal{R}\) to show that a fair sequence of rule applications exists. Second, in assigning ground terms to the existential variables \(\{!y_i\}\), we needed a denumerable set of individual constants \(\{c_i\}\) that did not appear in the query \(P(x)\), or in the initial substate \(s_0\), or in the rules \(\mathcal{R}\).

The first use of the finiteness assumption can be eliminated by a standard device. Assume, in the worst case, that the execution of every rule in the proof procedure creates a denumerably infinite set of applicable new rules. We can write these rules down as follows:

\[
\begin{align*}
R_1^1 & R_1^2 R_1^3 \ldots \\
R_2^1 & R_2^2 R_2^3 \ldots \\
R_3^1 & R_3^2 R_3^3 \ldots \\
\vdots & \vdots \\
R_k^1 & R_{k-1}^2, \ldots, R_2^{k-1}, R_1^k \\
\vdots & 
\end{align*}
\]

assuming that the execution of step \(S_k\) creates the new rules \(R_1^k, R_2^k, R_3^k, \ldots\). We can then execute these rules in the following order:

\[
R_1^1, \\
R_2^1, R_1^2, \\
R_3^1, R_2^2, R_1^3, \\
\vdots \\
R_k^1, R_{k-1}^2, \ldots, R_2^{k-1}, R_1^k, \\
\vdots 
\]

Under this arrangement, any rule that becomes applicable at step \(S_k\) will be executed after only a finite number of additional steps. Thus every component of the proof of Theorem 4.6 that depends on the existence of a fair sequence of rule applications remains valid for an infinite \(s_0\) or an infinite \(\mathcal{R}\).

The second use of the finiteness assumption cannot be eliminated, but must be incorporated as one of the constraints on \(s_0\) and \(\mathcal{R}\). The most natural way to do this is as follows:

**Theorem 4.14.** Let \(s_0\) be a (possibly infinite) initial substate in \(B\), and let \(\mathcal{R}\) be a (possibly infinite) set of rules including negations and embedded implications. Assume that all the individual constants in \(\mathcal{R}\) appear in \(s_0\), and assume that there exists an infinite set of individual constants in \(U\) that do not appear in \(s_0\). Then, if \(K^*\) is nonempty,

\[
\bigcap K^* = \bigcap J_\omega = \bigcup \{ \bigcap T^k_N (J_0) | k < \omega \}.
\]
Proof. By assumption, there is an infinite set of individual constants in \( U \) that do not appear in \( s_0 \) or \( S \). Thus the proof of Theorem 4.6 (completeness) goes through without modification, using the fair sequence of rule applications outlined above. Thus the proofs of Corollary 4.12 and Corollary 4.13 go through as well. \( \square \)

The constraint in Theorem 4.14 now suggests a slightly different approach to the definition of \( K^* \). Suppose we are given some arbitrary initial substate \( s_0 \), either finite or infinite, which includes the individual constants in \( S \). Choose a denumerable set of individual constants that do not appear in \( s_0 \), add these to the constants that do appear in \( s_0 \), and then construct the sets \( U \) and \( H \) and the lattice \([B, \leq]\) exactly as in Section 2 (Paper I). Finally, define \( K^* \) as the greatest fixed point of the transformation \( T(J) = T_s(J) \cap J_0 \), exactly as in Theorem 3.7. We are then guaranteed that the equality in Theorem 4.14 holds.

This construction of \( K^* \) is reminiscent of the construction of initial models in Horn-clause logic [10,11]. Translated into our present notation, the substate \( g^* \) is an initial model for the rules \( P \Leftarrow \bigwedge Q_j \) and \( P \Leftarrow \bigvee Q_j \) because there exists a unique homomorphism, preserving function and predicate symbols, from \( g^* \) into any other model that satisfies these rules. For clausal intuitionistic logic, where \( K^* \) is the "largest" Kripke model rather than the "smallest" Herbrand model, the analogous concept would be that of a unique homomorphism from any Kripke model of the rules into the particular model \( K^* \). It appears that \( K^* \) could be characterized as an initial Kripke model in this sense, given an appropriate definition of the homomorphism. This topic is beyond the scope of the present article, but it deserves further investigation.

4.6. Variations

We introduced the simplified proof procedure in Section 4.2 in order to develop a more tractable version of the soundness and completeness theorems. But we can now show that the simplified proof procedure and the standard proof procedure are equivalent, in the following sense:

Lemma 4.15. Any closed system of tableaux constructed according to the simplified proof procedure can be transformed into a closed system of tableaux constructed according to the standard proof procedure, and vice versa.

Proof. The proof is straightforward, but tedious, and we will only provide a sketch of it here. To see how to fill in the details, the reader should compare Figure 1 (the standard proof procedure) with Figure 3 (the simplified proof procedure).

Assume that we have been given a closed system of tableaux with an answer substitution \( \sigma \) constructed according to the simplified proof procedure. For simplicity, we will identify the tableau dependency tree for \( \sigma \) and ignore everything else. There are two kinds of transformations that have to be made: First, every principal refutation tree constructed from a rule \( F \Leftarrow P(x) \land Q(x; y) \) must be split into a subordinate refutation tree beginning with \( P(x) \) and having an answer substitution \( \sigma \) at its top node, plus a specialized rule \( Q(x; y)\sigma \Rightarrow F \). Second, every closed auxiliary tableau that returns an answer formula \( P(x)\sigma \) to a node containing \( Q(x; y)\sigma \) must be converted into an auxiliary tableau that is specialized by the
unification of $P(x)$ and $Q(x; y)\sigma$. In both cases, since the unifying compositions exist in the simplified proof, we are guaranteed that the specializations will go through in the standard proof, and that the final answer substitution will be the same.

Conversely, assume that we have been given a closed system of tableaux with an answer substitution $\sigma$ constructed according to the standard proof procedure. There are two kinds of transformations to be made here, reversing the transformations in the previous paragraph. First, every subordinate refutation tree for the formula $P(x)$ with answer substitution $\sigma$ must be spliced into the left branch of the appropriate tree $F = Q(x; y)\sigma$, and the branch beginning with $Q(x; y)\sigma$ must then be generalized into a branch beginning with $Q(x; y)$. Second, every closed auxiliary tableau for the rules $P \leftarrow Q$ or $P \leftarrow [Q \Rightarrow R]$ attached to a node containing $Q(x; y)\sigma$ must be generalized to an auxiliary tableau for the uninstantiated formula $P(x)$. In both cases, these generalizations must be "pushed" to the tips of the tableau dependency tree, so that a generalized answer substitution is returned. (The procedure is similar to the "inversion" procedure in Lemma 4.8.) Since a specialized answer substitution was returned to these nodes in the standard proof, by assumption, we are guaranteed that the unifying compositions will go through in the simplified proof, and that the final answer substitution will be the same. \[\square\]

Applying Lemma 4.15, we see that Theorem 4.3 (soundness) and Theorem 4.6 (completeness), which were formulated for the simplified proof procedure, are valid for the standard proof procedure as well.

Furthermore, once we understand how to transform closed systems of tableaux in this way, it is obvious that there are many other possible variations in the proof procedures. We list here several examples:

1. In Section 3.1, we showed that the expansion rules in the form $P \leftarrow [Q \Rightarrow R]$ could be eliminated from our system, thus simplifying the semantics. But it is now easy to see that these rules can be added back to the proof procedure, if desired. We simply set up a subordinate refutation tree in $\mathcal{T}$ beginning with $P(x)$, and, whenever, the tree closes with an answer substitution $\sigma$, we add the rule $Q(x; y)\sigma = R(x; y)\sigma$ to the rule base of $\mathcal{T}$. The transformation here is thus identical to the transformation from the simplified proof procedure using the rule $P \leftarrow [Q \Rightarrow R]$ to the standard proof procedure using the rule $P(x) = [Q(x; y) \Rightarrow F]$.

2. We also showed in Section 3.1 that the abstraction rules $P \leftarrow [Q \Rightarrow R]$ could be simplified by assuming that the variables $x$ and $y$ appearing in $R(x; y)$ also appear in $Q(x; y)$. This assumption can also be eliminated in our proof procedure, if desired. A principal refutation tree in $\mathcal{T}$ beginning with $R(x; y)$ could thus contain any number of extra variables $x$ and $y$, and these variables would be treated like any other variables in a Horn-clause refutation tree.

3. Finally, although we have used And/Or refutation trees throughout the present article to simplify the theoretical analysis, the proofs themselves can be constructed from SLD refutation trees, whenever desired. To transform a proof from an And/Or tree to an SLD tree, we simply specialize the right branches; to transform a proof from an SLD tree to an And/Or tree, we
generalize the right branches and compute the unifying compositions at the 'And' nodes. An SLD proof procedure does not always generate a fair sequence of rule applications, however, and so Theorem 4.6 (completeness) must be stated with some care. The situation is exactly the same in the pure Horn-clause case [1].

With such an array of possible proof procedures available for clausal intuitionistic logic, all of them sound and complete, we can begin to investigate questions of efficiency. Although most of these questions are beyond the scope of the present article, we will outline here one important heuristic device. The major source of inefficiency in the proof procedures considered so far arises from the negation rules $P \Rightarrow \neg Q$ and $P \Leftarrow \neg Q$. In the standard proof procedure, for example, the subordinate refutation trees for the rules $P \Rightarrow \neg Q$ can be constructed in any existing tableau, and the principal refutation trees beginning with $F$ must then be expanded in every auxiliary tableau, since any of these negation rules might lead to closure. But consider Example 4.1 again. Suppose we rename the negated predicates in this example, as follows:

$P_1(x) = \text{not-}Q_1(x, y),$
\[ \text{not-}Q_2(x, y) = P_2(x), \]
\[ \text{not-}Q_1(x, y) = \text{not-not-}R(x, y, z), \]
\[ \text{not-not-}R(x, y, z) = \text{not-}Q_2(x, y). \]

We have now constructed a simple set of Horn clauses, and we can see that $'P_1(a)'$ is computable from the initial substate $s_0 = \{P_2(a)\}$ by a straightforward Horn-clause refutation proof. Furthermore, the successful path through the Horn-clause refutation tree matches the path in Figure 1 from $\mathcal{T}_0$ to $\mathcal{T}_1$ to $\mathcal{T}_2$ and back again. This device of renaming predicates is a familiar trick among PROLOG programmers. For example, several of the solutions to the 'Alpine Club' puzzle posted on the PROLOG Digest mailing list (see also [25, p. 270], for a statement of this puzzle) worked essentially by renaming negations as affirmative 'not-' predicates.

Although the device of renaming predicates happens to give the correct solution to Example 4.1, it is not a sound inference procedure. Consider a much simpler set of rules:

$P_1(x) = \neg Q(x, y),$
$P_2(x) = \neg Q(x, y),$

where the device of renaming predicates also gives the correct solution. If we replace these rules with a set of rules including function symbols:

$P_1(x) = \neg Q(x, y),$
$P_2(x) = \neg Q(x, f(y)),$

we can easily see that $'P_1(a)'$ is not entailed in our system of clausal intuitionistic logic by the initial substate $s_0 = \{P_2(a)\}$. However, in the renamed version of these rules, $'P_1(a)'$ would be provable. Similarly, if we modify our initial set of rules by interposing a Horn clause:

$P_1(x) = \neg Q_1(x, y),$
$P_2(x) = \neg Q(x, y),$
$Q(x, y) = Q_1(x, y) \wedge Q_2(x, y),$
it is also obvious that \( P_i(a) \) is not entailed by the initial substate \( s_0 = \{ P_i(a) \} \).

In this latter case, we have to decide how to treat the Horn clause in the renamed version. Suppose we wrote this rule (incorrectly) as two Horn clauses:

\[
\text{not-}Q_1(x, y) \leftarrow \text{not-}Q(x, y) \quad \text{and} \quad \text{not-}Q_2(x, y) \leftarrow \text{not-}Q(x, y).
\]

Then '\( P_i(a) \)' would be provable, using the first of these rules. From these examples, then, it is clear that the device of renaming predicates cannot serve as a substitute for a full tableau proof in the system of clausal intuitionistic logic.

As a heuristic, however, the device has some merit. The existence of a renamed solution turns out to be a necessary (but not sufficient) condition for a proof. In other words, whenever there exists a correct proof in clausal intuitionistic logic, we can show that there exists a Horn-clause proof in the renamed version of the rules. Furthermore, the trace of the renamed proof matches the trace of the correct proof, through all the possible principal and subordinate refutation trees, in all the possible auxiliary tableaux that can be generated by the original rules. We can thus use this trace as a heuristic to control the generation of refutation trees from the negation rules \( P \rightarrow Q \) and \( P \leftarrow Q \). Looking at Figure 1 again, it should be obvious why this is so. Instead of searching \textit{top down} from \( F \) in the tableau \( T_1 \), which is often inefficient, the trace of the renamed proof specifies a \textit{bottom up} search from the data base of \( T_1 \). It remains to be seen, though, how successful this search heuristic would be in a realistic set of rules. This is a topic for future research.

5. RELATED WORK

There are two other similar approaches to intuitionistic logic programming in the current literature, by Dale Miller at the University of Pennsylvania [23] and by Dov Gabbay and his colleagues at Imperial College [7-9], both developed contemporaneously with the present system. In this section, we will survey these two projects, and mention briefly several additional projects that are also related.

The most closely related work is by Dale Miller [23]. Miller investigates a logic-programming language that includes embedded implications; he develops a fixed-point semantics for his language; and he shows that the resulting system implements a subset of intuitionistic logic. For a given program \( \mathcal{P} \) and a goal \( G \), the “operational semantics” of Miller’s language, denoted by \( \mathcal{P} \vdash G \), resembles a simplified version of the tableau proof procedure in the present article. Although Miller’s fixed-point operator initially seems to be quite different from the one developed here, it eventually computes a least fixed point that is equivalent to the substate \( \neg \neg \mathcal{K}^* \). The details are interesting: Starting with the set \( \mathcal{W} \) of all programs, Miller defines an \textit{interpretation} to be any function \( I: \mathcal{W} \rightarrow \mathcal{B} \) subject to a local monotone condition. (Here \( \mathcal{B} \) is the powerset of \( \mathcal{H} \), as in the present article.) He then constructs a complete lattice of interpretations, with a smallest element \( I_\bot \) defined by setting \( I_\bot(w) = \emptyset \) for all \( w \in \mathcal{W} \). Next, he defines recursively a weak notion of satisfaction: \( I, w \models G \), and he uses this definition to construct an operator \( T \) on the lattice of all interpretations. It turns out that \( T \) is both monotone and continuous, by virtue of the restricted syntax of the language, and thus the least fixed point of \( T \) is given by \( T^\infty(I_\bot) \). Finally, for any program \( \mathcal{P} \) and any closed goal formula \( G \), Miller shows that \( \mathcal{P} \vdash G \) if and only if \( T^\infty(I_\bot), \mathcal{P} \models G \). In other words, the operational semantics is equivalent to the least-fixed-point semantics, exactly as in the pure Horn-clause case.
In some respects, Miller's system is more general than the system in the present article, and in some respects it is more restricted. For example, Miller's notion of an interpretation is very general, since it takes into account the set of all possible programs, not just the set of all possible substates. A program $\mathcal{P}$ in Miller's system is essentially equivalent to $s_0 \sqcup \mathcal{P}$ in our system, and we have been working with the set of all substates $s' \geq s_0$ for some fixed $\mathcal{P}$, whereas Miller is effectively working with the set of all $\mathcal{P}' \geq \mathcal{P}$ as well. When we actually arrive at the computation of a fixed point, however, this additional generality may not be that significant. Applying Miller's least fixed point $T^\omega(I_{\bot})$ to the program $\mathcal{P} = s_0 \sqcup \mathcal{P}$, we can show that

$$T^\omega(I_{\bot})(s_0 \sqcup \mathcal{P}) = \mathcal{K}^\omega.$$

The most significant difference, then, may lie in the two alternative routes to essentially the same result. Miller's fixed-point construction is more abstract and homogeneous, treating all the rules alike. The fixed-point construction in the present article is more concrete, and more particularized to the specific types of rules. It would be interesting to see if there are any substantive differences arising from these alternative approaches.

In addition to these differences in the details of the fixed-point construction, there are also differences in the expressive power of the two systems, at least in the version that appears in Miller's published paper [23]. Miller's system is less expressive than the present system, since it does not permit the appearance of a universally quantified $\mathcal{y}$ variable in the rules $P \leftarrow \neg Q$ and $P \leftarrow [Q \Rightarrow R]$. This restriction simplifies the analysis substantially. For example, Miller is able to prove the continuity of the fixed-point operator $T$ directly in his paper, but his proof, as given, would not go through if there were universally quantified variables on the right-hand side of the embedded implication rules. (Recall that the failure of continuity in the counterexamples in Section 3.4 resulted from the presence of the variable $\mathcal{y}$ in the rule $P(x) \leftarrow \neg Q(x, \mathcal{y})$.) Thus, a different approach would be necessary in this case. Miller has recently reported (personal communication) that he has extended his language to incorporate universally quantified $\mathcal{y}$ variables in the rules $P \leftarrow \neg Q$ and $P \leftarrow [Q \Rightarrow R]$, and that he has constructed a continuous fixed-point semantics for this extension. It would be interesting to compare this extended system with the system in the present article.

Although Miller presents several instructive examples of his operational semantics, he does not specify a general proof procedure for his logic. In fact, as he points out, the "proof rules" in the operational semantics would require an interpreter to "guess" at a closed term, thus causing a potentially infinite branching. The obvious solution to this problem is to replace these "proof rules" with some form of unification. But how should the free variables be treated? This has been one of the main concerns in the work of Dov Gabbay and his colleagues [7–9]. In [7], Gabbay and Reyle discuss several possible proof procedures for the languages $\mathcal{N}$-PROLOG and $\mathcal{QN}$-PROLOG, and for intuitionistic logic programming in general. Their strategy is to specify the propositional system first, and then extend it to incorporate quantifiers. $\mathcal{N}$-PROLOG is thus defined as a language that includes embedded implications, but excludes negation, and is restricted to the propositional case. The computation rules for this language are relatively simple. For example, the rule for embedded implications is exactly the same as Miller's rule: $\mathcal{P} \vdash_0 A \Rightarrow B$ if and only if $P \cup \{A\} \vdash_0 B$. The quantificational language $\mathcal{QN}$-PROLOG is then defined by
replacing the atomic propositions in N-PROLOG with atomic formulae, which are constructed from function symbols and predicate symbols in the usual way. The computation rules are also generalized to require unifying substitutions for the free variables, and to define the "joint success" of a set of queries \{\mathcal{P}, \vdash_\theta G_i\} for a single substitution \theta.

However, in order to develop a proof procedure for QN-PROLOG, Gabbay and Reyle are forced to impose several severe restrictions on the syntax of their language. First, the appearance of a universally quantified \(y\) variable on the right-hand side of a rule \(P \leftarrow [Q \Rightarrow R]\) is specifically disallowed, so that QN-PROLOG is restricted initially to the same expressive power as the language considered by Miller. Second, the computation rules of QN-PROLOG depend on a distinction between two classes of variables: The variables in \(\text{VAR}_1\) are universally quantified, and the variables in \(\text{VAR}_2\) are existentially quantified. But the variables in \(\text{VAR}_1\) are not allowed to appear in complex terms, so it is impossible to write a great many naturally occurring clauses containing universal quantifiers. For example, it appears that the set of rules in Example 4.2 in Section 4.1 of the present article would be syntactically excluded from QN-PROLOG. Finally, in order to discuss the implementation of QN-PROLOG in PROLOG itself, either at the object level or at the meta level, Gabbay and Reyle impose a further restriction: \(\text{VAR}_1\) and \(\text{VAR}_2\) variables may not appear together in the same clause. As we have seen in the present article, however, these restrictions are unnecessary. Using the technical device of the local tableau variables ?x, our proof procedure solves these quantificational problems without imposing any restrictions on the form of the rules. And using the technical device of the existential variables !y, our proof procedure is able to handle an extended set of rules \(P \leftarrow [Q \Rightarrow R]\) with universal quantifiers embedded on the right-hand side.

In a sequel to this initial article, Gabbay investigates the logical foundations of N-PROLOG and QN-PROLOG [8]. As in the present article, the semantics of the language is intuitionistic. Gabbay thus proves the completeness of propositional N-PROLOG by explicitly constructing an intuitionistic Kripke model, and he generalizes this result to QN-PROLOG by analyzing the "propositional freeze" of a successful computation. The soundness-and-completeness theorem is stated as follows: \(\mathcal{P} \vdash_\theta G\) in QN-PROLOG if and only if

\[
\vdash_f (\exists y_2 \in \text{VAR}_2) [ (\forall y_1 \in \text{VAR}_1) \mathcal{P} \Rightarrow G ],
\]

where "\(\vdash_f\)" denotes intuitionistic provability. A critical step in the proof of the completeness theorem depends on the existential property of intuitionistic logic, namely, that if \(\vdash_f (\exists x) \mathcal{A}(x)\) for some well-formed formula \(\mathcal{A}\), then there exists a ground substitution \(\theta\) into the Herbrand universe such that \(\vdash_f \mathcal{A}(x)\theta\). Because of this property, Gabbay's proof can proceed from the existentially quantified statement in (13) to a particular ground substitution \(\theta\) that renders (13) true, and from there to a propositional trace of the computation rules in QN-PROLOG. Notice that Gabbay does not state the completeness theorem for QN-PROLOG in terms of the following implication:

\[
\vdash_f (\forall y_1 \subseteq \text{VAR}_1) \mathcal{P} \Rightarrow (\exists y_2 \subseteq \text{VAR}_2) G,
\]

or equivalently,

\[
(\forall y_1 \subseteq \text{VAR}_1) \mathcal{P} \vdash_f (\exists y_2 \subseteq \text{VAR}_2) G,
\]
since (14) and (15) are not intuitionistically equivalent to (13) for an arbitrary first-order formula $\mathcal{P}$, even if $\mathcal{P}$ contains no free variables in $\text{var}_2$. In the present article, however, where the program $\mathcal{P}$ is restricted to rules of the form $P = \land Q_j$, $P \equiv \lor Q_j$, $P \equiv \neg Q$, and $P \equiv \{Q \Rightarrow R\}$, and where the goal $G$ is restricted to an atomic formula $P(x)$, we have actually established a stronger version of the completeness theorem, closer to (15) than to (13). Notice that the concept of entailment in Definition 2.2 is equivalent to $s_0, \mathcal{R} \models \exists x \ P(x)$, and that the concept of uniform entailment in Definition 2.3 is equivalent to $s_0, \mathcal{R} \models \exists x \ P(x) \theta$ for some ground substitution $\theta$. Corollary 3.16 to Theorem 3.15 then tells us that $(\exists x) P(x)$ is uniformly entailed by $s_0$ and $\mathcal{R}$ whenever $(\exists x) P(x)$ is entailed by $s_0$ and $\mathcal{R}$. Thus, although Theorem 4.6 (completeness) was stated in terms of uniform entailment, or $s_0, \mathcal{R} \models \exists x \ P(x) \theta$ for some ground substitution $\theta$, it could also be stated in terms of entailment, or $s_0, \mathcal{R} \models \exists x \ P(x)$. Obviously, this would not be the case if the rules $\mathcal{R}$ included arbitrary first-order formulae. For example, since the disjunctive assertion 'P(a) $\lor$ P(b)' does not generate a unique minimal substate, Corollary 3.16 would not be true for a program that included this assertion as its only rule.

Although Gabbay does not include negation rules in the form $P = \neg Q$ or $P \equiv \neg Q$ in his presentation of n-PROLOG and QN-PROLOG, a related article by Gabbay and Sergot [9] introduces negation into logic programming in a similar way. Their basic idea is to augment a positive program $\mathcal{P}$ with a set of negative clauses $\mathcal{N}$, and then attempt to prove both positive goals, $G$, and negative goals, $\neg G$, from the pair $(\mathcal{P}, \mathcal{N})$. Notice that the negative clauses in $\mathcal{N}$ can be represented by our negation rules $P \equiv \neg Q$, and the negative goals $\neg G$ can be represented by our negation rules $P \equiv \neg Q$, so these two systems are fully equivalent. In fact, Gabbay and Sergot establish some of the properties of their system by translating it into n-PROLOG and interpreting a negated formula $\neg G$ as a rule $G \Rightarrow F$, exactly as we have done in our tableau proof procedures. Given this translation, Theorem 4.3 (soundness) and Theorem 4.6 (completeness) could be used to establish the soundness and completeness of Gabbay and Sergot’s system, relative to an intuitionistic semantics. Curiously, Gabbay and Sergot do not discuss their system in terms of intuitionistic logic, but instead compare it to classical logic. They thus prove that both the propositional and the quantificational versions of their system are sound with respect to classical logic, which is a weaker conclusion than soundness with respect to intuitionistic logic. They also prove, for the propositional case, that if $G$ is classically provable from $(X \mid X \in \mathcal{P}) \cup (\neg Y \mid Y \in \mathcal{N})$, then the goal $\neg G$ will succeed in their system from the program $(\mathcal{P}, \mathcal{N})$. Gabbay and Sergot do not state a completeness theorem in the quantificational case, however, and it appears that a proper theorem here would have to be formulated in terms of intuitionistic logic.

The languages investigated by Miller and Gabbay differ syntactically from the language investigated in the present article, but only in a superficial sense. Miller’s language and Gabbay’s language both permit embedded implications (and negations) to be nested to an arbitrary depth, whereas the system of clausal intuitionistic logic does not. However, we can always achieve the effect of indefinite nesting by defining new predicates $P_1 \equiv \{Q \Rightarrow R\}$ for the nested implications, or $P_1 \equiv \{\neg Q\}$ for the nested negations. We have thus opted for syntactic simplicity in the rules, without sacrificing semantic expressiveness. To some extent, these syntactic differences reflect a difference in the initial motivation for the work. The system of clausal intuitionistic logic was originally developed for the purposes of knowledge representation and common-sense reasoning, prompted by earlier work on the representa-
tion of knowledge in legal domains [21,17], and the rules in these domains tend to use a great many explicit definitions, with only minimal nesting. By contrast, Miller's work has been motivated primarily by programming examples, where deep nesting is common. Miller thus shows how embedded implications can be used to construct modules in logic-programming languages, and he presents several examples of the use of his system to develop logically correct implementations of certain desirable side effects. Similarly, Gabbay and Reyle present several examples of the use of N-PROLOG to name clauses in a data base, and to control the search order of a program without the use of extralogical devices. It is interesting to note, however, that Gabbay and Reyle also provide an initial practical motivation for their system by discussing the formalization of the British Nationality Act, a legal example, and Gabbay and Sergot likewise motivate their analysis of negation by discussing several legal examples. Perhaps there is a common element in the legal examples and the programming examples that somehow stimulates the development of intuitionistic logics.

In addition to the articles by Miller and Gabbay on intuitionistic logic programming, there is an early paper by David S. Warren [30] that investigates the problem of data-base updates in PROLOG, and proposes a Kripke semantics for an operation that resembles the operation of our embedded implication rules. There are also several papers in the current literature that are similar in motivation, although not explicitly based on intuitionistic logic. For example, Peter Patel-Schneider has been interested in the development of knowledge-representation languages with tractable computational properties, and he has proposed a variant of first-order four-valued relevance logic for this purpose [26]. Although first-order relevance logic is itself undecidable, Patel-Schneider shows that he can obtain a decidable system by further restricting the interpretation of the existential quantifier to its "intuitionistic reading". This does not produce an intuitionistic logic, however, but a much weaker logic, which lacks the rule of modus ponens.

The idea of using partial models as the foundation for a theory of logic programming has been thoroughly investigated by Melvin Fitting [6]. Although Fitting's system is classical, rather than intuitionistic, by using partial models he is able to define three truth values for a logic program: true, false, and unknown, to represent a program with infinite loops. Fitting also develops a powerful fixed-point semantics for his logic, generalizing the fixed-point semantics of Apt and van Emde [1] in a different way than we have generalized it in the present article. More abstractly, the notion of a partial model has appeared in several proposals for the representation of common sense knowledge. Although the precise details are different in each case, the use of partial models is a prominent feature in the work of Barwise and Perry [2] and in the work of Hans Kamp [12], to list just two recent examples.

6. CONCLUSION

We have investigated in this article one possible way to extend the expressive power of Horn-clause logic: Add negations and embedded implications to the right-hand side of a rule, and interpret these new rules intuitionistically, in a set of partial models. We have stated and proven several theorems about this extended system, and we have provided several examples of its operation. But there is also a stronger thesis here, appearing implicitly throughout the article. We are suggesting, in fact,
that an intuitionistic semantics provides the "correct" way to augment our existing systems of logic programming, both conceptually and computationally. To this end, we have emphasized in this article the similarities between the theory of clausal intuitionistic logic and the classical theory of Horn clause logic [29,1], specifically, the existence of a fixed-point semantics that resembles that of Horn clauses, and the existence of a tableau proof procedure that generalizes Horn-clause refutation proofs. But the thesis is controversial, and there is ample room for disagreement.

Although various other researchers have been exploring similar ideas as discussed in Section 5, there are several advantages to the system of clausal intuitionistic logic developed here. First, our system includes both negation rules and embedded implication rules, and it treats these rules uniformly in both the model theory and the proof theory. Second, our system permits the appearance of a universally quantified \( y \) variable on the right-hand side of a rule \( P \leftarrow \neg Q \) or \( P \leftarrow [Q \Rightarrow R] \), thus enabling us to handle many common examples, such as the expressions (1) and (3) in Section 1, where these universally quantified variables are necessary. Third, since our tableau proof procedures are based on a general unification mechanism that works with two special kinds of variables, \(?x\) and \(!y\), we have been able to avoid the introduction of ad hoc devices in the proofs without artificially restricting the syntax of the rules. Finally, since the systems of tableaux constructed in the proofs correspond directly to the successive applications of \( T_N \) to \( J_0 \) in the semantics, the soundness and completeness theorems for clausal intuitionistic logic are relatively simple and clean. This last point is also one of the major features of the classical theory of Horn-clause logic programming [1].

In addition to providing a contribution to the theory of logic programming, the system of clausal intuitionistic logic is intended to serve as a foundation for further work on knowledge representation and common-sense reasoning. Some of this work was outlined briefly in [18], a paper discussing the relationship between intuitionistic logic and nonmonotonic reasoning [22,16,27]. The basic idea here is to view nonmonotonic reasoning as a computation in a single partial model, so that a nonmonotonic inference is always an unsound approximation to the correct inference in a system of intuitionistic logic. Several familiar forms of nonmonotonic reasoning can be explained in this way. For example, "negation as failure" [5] can be explained as an approximation to intuitionistic negation, and "domain circumscription" [16] can be explained as an approximation to intuitionistic implication with embedded universal quantifiers. Our original examples of an unowned property and a sterile container in Section 1 (Paper I) illustrate these relationships. In addition, a much broader class of nonmonotonic phenomena can be explained in this way by adding a new type of rule to our system. Suppose we reverse the direction of the implication in the rules \( P \leftarrow \land Q_j \) and \( P \leftarrow \lor Q_j \), and write down a set of rules in the form

\[
P(x) \Rightarrow Q_1(x;y) \land Q_2(x;y) \land \cdots \land Q_n(x;y)
\]

and

\[
P(x) \Rightarrow Q_1(x;y) \lor Q_2(x;y) \lor \cdots \lor Q_n(x;y),
\]

where the variable \( y \) on the right-hand side is interpreted existentially. If we now add these disjunctive expansion rules to the rules in the present article, we will obtain a system that is (intuitionistically) equivalent to a full first-order (intuitionistic) logic. For this extended system, we can show that the fixed-point construction in
Theorem 3.7 still goes through, but of course the set $K^*$ does not have a unique minimal substate $\models K^*$, and the tableau proof procedure does not work properly. In this situation, however, we can designate one of the disjuncts in the rule (17) as prototypical, and we can compute with the tableau proof procedure in the prototypical partial model, thereby obtaining an unsound approximation to a query that is otherwise too complex to compute correctly. These ideas then lead to a rigorous analysis of prototypes and deformations, a representational formalism that was developed in our earlier work [21] on purely empirical grounds. The details of this analysis will be presented in a forthcoming paper [20], which is the complete version of [18].

The techniques of clausal intuitionistic logic, although restricted in the present article to a first-order nonmodal language, are also applicable to a wide variety of more complex languages, including various languages for the representation of events, and actions, and the common-sense modalities over actions. An example of such a language was presented in an earlier paper on "Permissions and Obligations" [17]. Three distinct language levels were constructed in that paper: a first-order language $L_1$ defined on states, an action language $L_\alpha$ defined on sequences of states, or worlds, and a deontic language $L_D$ defined on sets of worlds. Significantly, the meaning of the expressions in each of these languages was specified in terms of partial models, called substates and subworlds, exactly as in the present article, and the use of partial models in the semantics turned out to be essential for the correct representation of a "free choice permission." However, the overall logic of the system was classical, rather than intuitionistic, and this fact caused some unnecessary complications. For example, the earlier paper did not attempt to develop a proof procedure for the language of permissions and obligations, and it suggested that the proofs might be intolerably complex for certain kinds of expressions in the action language. It now appears that many of these complexities can be avoided by the explicit use of an intuitionistic semantics for the languages $L_1$ and $L_\alpha$, and by the use of a tableau proof procedure for queries in the deontic language $L_D$. These ideas are discussed, without the full technical details, but with the use of an extended example, in [19]. The technical details will be presented in a future paper.

This research was partially supported by the National Science Foundation under Grant MCS-82-02591, and partially supported by the David Sarnoff Research Center in Princeton, New Jersey. The author wishes to thank Tony Bonner, Fritz Henglein, Tomasz Imielinski, and Dale Miller for their many helpful discussions as the article was being written. Special thanks are also due to Logan McCarty for his assistance with the illustrations.

REFERENCES