A COMPARISON OF THREE PROLOG EXTENSIONS*

DAVID W. REED AND DONALD W. LOVELAND

We present three extensions of PROLOG, discussing their similarities and differences. The systems—near-Horn PROLOG (Loveland), simplified problem reduction format (Plaisted), and N-PROLOG (Gabbay and Reyle)—differ in their approach to the extension of PROLOG, yet each utilizes case analysis as the mechanism for non-Horn reasoning. The fact that these systems, with quite different origins, purposes, and presentation forms, utilize the case-analysis method in a strikingly similar fashion suggests that their underlying reasoning is general and intuitive. This paper describes the three systems and outlines the close relationship between them. The systems also appear to have essential differences: properties of one system that cannot be incorporated in another without serious distortion of the unique properties of the receiving system. Highlighting those tradeoffs aids our understanding of the systems.

1. INTRODUCTION

One of the reasons for the popularity of the logic-programming language PROLOG is its well-understood basis in Horn-clause logic. An active area of research in recent years has been in extending PROLOG to broader classes of logic. We present three extensions of PROLOG, discussing their similarities and differences. The systems—near-Horn PROLOG (Loveland), simplified problem reduction format (Plaisted), and N-PROLOG (Gabbay and Reyle)—differ in their approach to the extension of PROLOG, yet each utilizes case analysis as the mechanism for non-Horn reasoning. Each of the three systems has received or is receiving extensive study and treatment in the literature.

*This research was partially supported by NSF Grant IRI-8805696 and ARO Grant DAAL03-88-K-0082.

Address correspondence to David W. Reed, Department of Computer Science, Duke University, Durham, NC 27706.

Received June 1989; accepted December 1989.

THE JOURNAL OF LOGIC PROGRAMMING


655 Avenue of the Americas, New York, NY 10010 0743-1066/92/$3.50
Near-Horn PROLOG extends PROLOG to the full first-order logic while retaining much of the form and flavor of PROLOG as a programming language. The simplified problem reduction format also extends PROLOG to the full first-order logic, but it was designed more as a theorem proving system. N-PROLOG is basically an intuitionistic extension that adds hypothetical implication to PROLOG, but it can be seen to be classically complete on certain restricted input forms. This study of the relationships among these three PROLOG extensions is not to criticize their simultaneous development, but is to better understand the concepts that unite them and to highlight some apparently important differences. The fact that these systems, with quite different origins, purposes, and presentation forms, utilize the case-analysis method in a strikingly similar fashion suggests that there exists an important concept that needs to be exposed and better understood. There also appear to be essential differences: properties of one system that cannot be incorporated in another system without serious distortion of the unique properties of the receiving system. Highlighting those tradeoffs aids our understanding of the systems involved.

For the sake of simplicity, the systems will be presented at the propositional level, but all three lift to first-order clauses in the usual way. We will present the systems as refutation systems, introducing a new, reserved literal \textit{FALSE} to represent "absurdity". While near-Horn PROLOG and N-PROLOG were not originally presented as refutation systems, the conversion is an intuitive one. (In [5], Loveland formalizes the conversion for near-Horn PROLOG.)

Although a general description of near-Horn PROLOG is given, this paper focuses mainly on a variant of near-Horn PROLOG called \textit{Inheritance near-Horn PROLOG} (first described in [9], also presented in [6]). Inheritance near-Horn PROLOG is presented in Section 2 along with an example refutation. Section 3 similarly presents the simplified problem reduction format. A close variant of this system is presented which makes the comparison with Inheritance near-Horn PROLOG even more straightforward. Section 4 presents N-PROLOG and shows how restricting its input form allows the other systems to be embedded into it.

\section{2. nH-PROLOG}

The first system we present is \textit{near-Horn PROLOG} (nH-PROLOG), developed by Loveland [4,5,10]. The goal of nH-PROLOG is to extend PROLOG to full classical logic while remaining as close as possible to the flavor and form of PROLOG. We will give a general description of nH-PROLOG but mainly focus on a conceptually simpler variant called \textit{Inheritance near-Horn PROLOG} (InH-PROLOG) [9,6]. Depending on the implementation, InH-PROLOG may have a slower inner-loop speed than the original system, but it can produce shorter refutations and possibly limit the search space.

Throughout this paper, we adopt the Clocksin-Mellish [1] notation for PROLOG, which we assume to be familiar to the reader. An nH-PROLOG program is presented as one presents a PROLOG program, with two exceptions. The primary exception is that the head of a clause can have more than one literal: we call such a clause a \textit{multihead} or \textit{non-Horn} clause and separate the heads with semicolons representing disjunction. Secondly, all-negative clauses (clauses without heads) are
given the new, reserved literal FALSE as head. FALSE is intended to represent
"absurdity", so adding it as head does not alter the logical meaning of the clauses.
We will present nH-PROLOG as a refutation system, so the goal is to show that
FALSE follows from the program clauses.

We first give an intuitive overview of nH-PROLOG and the InH-PROLOG
variant. An nH-PROLOG derivation is a sequence $B_1, \ldots, B_n$ of blocks, with each
block resembling a full PROLOG derivation. The start block $B_1$ is a PROLOG
derivation of FALSE with the alteration that if a goal $H_i$ calls a non-Horn clause
with head literals $H_{i_1}, \ldots, H_{i_m}$, then the auxiliary heads $(H_{i_1}, \ldots, H_{i_{i-1}}, H_{i+1}, \ldots, H_{i_m})$
are simply ignored (deferred) for this block. That is, the called clause is treated as
if it were a Horn clause with head $H_i$. As in PROLOG, the block terminates when
the continuation is empty. However, here we must continue the computation in
order to remove the deferred heads. For each deferred head, there is a restart
block $B_i (i > 1)$ whose task is to remove that head. The deferred head is promoted
to distinguished active head, and the new operation of cancellation permits it to
satisfy (cancel) any matching goal occurrence in the restart block. The distin-
guished active head can be seen as a conditional fact that holds in that block and
can be used to satisfy goals. Except for the cancellation operation, a restart block
behaves the same as the start block, so reduction via a non-Horn clause can
introduce other deferred heads. If we obtain the empty continuation and have
incurred a cancellation (requiring this constitutes a strong pruning rule), then the
block is successful and the distinguished active head is deleted from further
consideration. Computation terminates when some block removes the last deferred
head. The pruning rule requiring that the distinguished active head be used within
the block is called the cancellation pruning rule.

InH-PROLOG differs from nH-PROLOG primarily by allowing more than one
active head in blocks, all with cancellation capability. In addition to the distin-
guished active head promoted from the deferred-head list, the restart block
"inherits" all of the active heads from the block in which the distinguished active
head was deferred. For example, if head $H$ was deferred in a block with active
heads $A_j, \ldots, A_1$, then the restart block which removes deferred head $H$ will have
active heads $H, A_j, \ldots, A_1$, with $H$ being the distinguished active head. (While we
will omit the details here, all of the information necessary for inheritance can be
localized to the previous block, assuming a stack like implementation of the
deferred-head list.) The removal of the distinguished active head is still consid-
ted the task of a restart block. Again, a strong cancellation pruning rule is enforced
that requires at least one cancellation by the distinguished active head within the
block (even when cancellation by other active heads occurs).

The basic idea here is to use case analysis to perform non-Horn reasoning. It
can be seen that $P \cup \{H_i; \ldots; H_m; \leftarrow L_1, \ldots, L_n\}$ (where $P$ is a set of clauses) is
unsatisfiable if $P \cup \{H_i; \leftarrow L_1, \ldots, L_n\}$ is unsatisfiable for some $i$ and $P \cup \{H_j,\}$ is
unsatisfiable for all $j \neq i$. As described above, the start block is a refutation of the
case $P \cup \{H_i; \leftarrow L_1, \ldots, L_n\}$, since the other heads are ignored. The restart blocks
are refutations of the cases of the form $P' \cup \{H_i,\}$, since $H_i$ is made distinguished
active head and can be used as a conditional fact. Hence, the input clauses are
broken into cases to obtain clause sets with one less non-Horn clause than $P'$. The
recursive use of this case analysis essentially breaks the input set into Horn set
cases, which are handled exactly as in PROLOG except for the new cancellation
The conjunction of these PROLOG-like refutations constitutes an InH-PROLOG refutation of the input set.

We can now make the general description above more precise. Each line of an InH-PROLOG refutation has the following format:

\[ C \# A \{ D \}, \]

where \( C \) is a sequence of goals (the continuation) separated by commas representing conjunction (with the identical connotation of a standard PROLOG continuation), \( A \) is a sequence of literals called active heads (the leftmost being the distinguished active head), and \( \{ D \} \) is a list of deferred heads that functions as a left-ended stack. The symbol ‘\#' is called the wall. By convention, the deferred-head list will not be explicitly written if it is empty, and the wall will not be written if there are no active or deferred heads.

An InH-PROLOG refutation is a finite sequence of such lines, with the following properties:

1. The first line is of the form ‘FALSE’. Note that there are no active or deferred heads.

2. If line \( n \) contains a goal, then the leftmost goal (by convention) is the calling goal.
   a. If the calling goal matches with an active head, then the calling goal is removed (canceled). Line \( n + 1 \) inherits the rest of line \( n \).
   b. If there are clauses with head matching the calling goal, choose one such clause and call it \( C \). Line \( n + 1 \) replaces the calling goal with the body of \( C \) and inherits the rest of line \( n \). If \( C \) is a non-Horn clause, each auxiliary head is added to the left of the deferred-head list.

3. If line \( n \) contains no goals, if the deferred head-list is nonempty, and if in the case of a restart block at least one cancellation by the distinguished active head has occurred within the block (the cancellation pruning rule), then a restart line follows. The restart line has the single goal ‘\#’ and inherits the deferred-head list from the last line of the previous block minus the leftmost deferred head, which becomes the distinguished active head. In addition, all of the active heads from the block in which the distinguished active head was deferred are inherited (added to the right of the distinguished active head).

4. A line containing no goals or deferred heads is the concluding line of an InH-PROLOG refutation. Like other restart blocks, the final restart block must obey the cancellation pruning rule.

To help in understanding the system, we now present an example refutation:

**Example 2.1.** Consider the following clause set:

\[
\begin{align*}
\text{FALSE} & : \neg h_1, y. \\
\text{FALSE} & : \neg h_2.
\end{align*}
\]

\[ y. \]

\[ h_1; h_2 : \neg y. \]
We have the following InH-PROLOG refutation of these clauses (the lines of the refutation are numbered, with a blank line separating the blocks):

1) \textit{FALSE}
2) \(h, y\)
3) \(y, y \# \{h_2\}\)
4) \(y \# \{h_2\}\)
5) \(# \{h_2\}\)
6) \textit{FALSE} \# \(h_2\)
7) \(h_2 \# \{h_2\}\)
8) \# \(h_2\)

Note that there are no active heads in the start block. Reduction [as described in step (2) (b)] proceeds as in PROLOG until the non-Horn clause matches with the calling goal \(h,\) in line 2. The calling goal is replaced with the literal \(y\) from the body of the clause, and the deferred head \(h_2\) is introduced in line 3. The block is then concluded via repeated reduction with fact \(y.\) Since the continuation is empty in line 5 but the deferred head list is not, a restart line follows [as described in step (3)]. The ensuing restart block has distinguished active head \(h_2\) (no active heads are inherited, since \(h_2\) was deferred in the start block), and a cancellation occurs in line 7 [as described in step (2) (a)], so the block concludes successfully.

The block structure of InH-PROLOG refutations highlights the case analysis being performed. In the start block of the above refutation, the non-Horn clause is treated as if it were a Horn clause with head \(h,\). This block is a refutation of the case \(P \cup \{h, : \sim y.\},\) where \(P\) represents the other clauses. In the restart block, the active head \(h_2\) is used as a conditional fact which can satisfy goals. This block is a refutation of the case \(P \cup \{h_2\}.\) Since the unsatisfiability of these two cases implies the unsatisfiability of the original clause set, the conjunction of these blocks represents a refutation of the original set.

\textbf{2.1. Declarative and Procedural Readings}

The above description presents an intuitive view of the case-analysis nature of InH-PROLOG. Because InH-PROLOG is an extension of the logic-programming language PROLOG, it is desirable to present more detailed declarative and procedural readings of a program and the system which processes the program. The declarative reading of each program clause \(H_1; \ldots; H_m : \sim L_1, \ldots, L_n,\) is, as expected, \((L_1 \land \cdots \land L_n) \rightarrow (H_1 \lor \cdots \lor H_m).\). Recall that in a PROLOG derivation, a continuation \(C\) can be read as showing \((P \land C^\lor) \rightarrow Q,\) where \(P\) is the program, \(C^\lor\) is the conjunction of literals of \(C,\) and \(Q\) is the original query. Similarly, a line in an InH-PROLOG derivation can be given a declarative reading.

A line \(C\#A[D]\) can be read as \((P \land \lnot D^\lor \land A^\land \land C^\lor) \rightarrow Q,\) where \(P\) is the program, \(D^\lor\) is the disjunction of deferred heads of \(D, A^\land\) and \(C^\lor\) are the conjunctions of literals of \(A\) and \(C\) respectively, and \(Q\) is the original query, here \textit{FALSE.} That is, if the deferred heads are all false (alternative cases do not apply) and all active heads and literals of \(C\) are true, then this together with the program
implies the original query. We can focus our attention on a particular block by taking $\neg D^\vee$ to be true, i.e. by disregarding the alternative cases which correspond to other blocks. The declarative reading of the line is then reduced to $(P \land A \land C) \rightarrow Q$ within the context of the block. This can be read as $(P' \land C^\vee) \rightarrow Q'$, the semantics for PROLOG with $P' \equiv P \land A$. This agrees with our view of deduction within a block as the standard logic-programming paradigm with conditional facts (the active heads) augmenting the given program $P$.

Regarding a procedural reading, the clause $H_1; \ldots; H_m \leftarrow L_1, \ldots, L_n$ is to be regarded as a multientry subroutine, callable by any $H_i$. This reading is valid locally, i.e. within a block. Globally, the remaining $H_i$ must be considered, unlike alternate entry points in standard procedures, but this is a property of blocks. Thus, the useful procedural viewpoint is to consider blocks as cases to be dismissed and the content of each block as the computation needed to dismiss the case. We thus say that InH-PROLOG has a local procedural reading.

3. SPRF

The next system we will consider is the simplified problem reduction format (SPRF) developed by Plaisted [8, 7]. The SPRF is a Gentzen-style axiom system that employs sequents of the form $\Gamma \rightarrow L$ where $\Gamma$ is a set of positive literals and $L$ is a positive literal. The axioms for the system are of the form $\Gamma \rightarrow L$ for $L \in \Gamma$.

The input set is expected to be in clausal form (as in nH-PROLOG) where all-negative clauses are given the reserved literal $FALSE$ as head. An inference rule is generated for each clause of the input as follows. To be precise, all of the inference rules listed here are inference-rule schemas, with variables ranging over the positive literals.

For each Horn clause $H :- L_1, \ldots, L_n$ there is the inference rule (where $\Gamma$ is an arbitrary set of positive literals)

$$
\Gamma \rightarrow L_1 \quad \cdots \quad \Gamma \rightarrow L_n \quad \Gamma \rightarrow H
$$

We may consider a fact to be a degenerate case of this form, where the body is empty. This results in an inference rule with no upper sequents, i.e. an axiom of the form $\Gamma \rightarrow H$ where $H$ is a fact.

For each non-Horn clause $H_1; \ldots; H_m :- L_1, \ldots, L_n$ there is the inference rule (where $\Gamma$ is an arbitrary set of positive literals and $U$ is a positive literal)

$$
\Gamma \rightarrow L_1 \quad \cdots \quad \Gamma \rightarrow L_n \quad \Gamma, H_i \rightarrow U \quad \cdots \quad \Gamma, H_m \rightarrow U \quad \Gamma \rightarrow U
$$

A rule of this form is known as a splitting rule. Plaisted notes that the splitting rule may be restricted to the single goal $FALSE$, i.e., the variable $U$ in the rule may be replaced by the literal $FALSE$. Since this restriction still yields a complete system, we will assume the restricted form for our comparison. We will later mention how the general form of the splitting rule alters the comparison.
The classical completeness and soundness of the SPRF are proven by Plaisted in [8, 7]. A set of clauses is unsatisfiable if and only if the sequent \( \Gamma \rightarrow FALSE \) can be derived from the axioms by the generated inference rules.

Intuitively, deriving the sequent \( \Gamma \rightarrow L \) can be interpreted as showing that “assuming the literals in \( \Gamma \), we are able to prove \( L \) from the clauses”. For this reason, we will refer to literals in the antecedent as assumptions and the literal in the succedent as the goal. Seen this way, the axioms \( \Gamma \rightarrow L \) for \( L \in \Gamma \) are obvious, since if \( L \) is in \( \Gamma \), we have proven \( L \) trivially by assuming it. The inference rules generated for Horn clauses match how PROLOG handles clauses—proving each goal in the body of a clause (given the assumptions in \( \Gamma \)) proves the head (given \( \Gamma \)). The splitting rules for non-Horn clauses are an extension to this, where case analysis is used to prove \( U \) by proving the goals in the body and also showing the \( U \) follows from each of the heads. (Note the similarity to the case analysis performed in InH-PROLOG.) Finally, given these interpretations, a derivation of \( \Gamma \rightarrow FALSE \) shows that “absurdity” can be proven from the clauses without making any assumptions, and thus is a refutation.

Example 3.1. Consider our example clauses and their corresponding SPRF inference rules (the rules are labeled to make refutations easier to follow—here Horn-rule labels start with \( H \) and splitting-rule labels start with \( S \)):

\[
\begin{align*}
FALSE & :- h_1, y. \\
FALSE & :- h_2. \\
y. & :- h_1, h_2. \\
h_1; h_2 & :- y.
\end{align*}
\]

These inference rules, along with the axioms, generate the following refutation:

\[
\begin{array}{c}
\Gamma \rightarrow y \\
\Gamma \rightarrow FALSE \\
\Gamma \rightarrow FALSE
\end{array}
\]

3.1. SPRF with Delay

The major drawback of SPRF is that when backchaining as in PROLOG, there is no guidance as to which splitting rules are needed or in what order. Although restricting splitting rules to have goal \( FALSE \) helps in controlling the search, splitting rules still must be used at the beginning of the backchaining refutation (when \( FALSE \) is goal), and at that point in the search there is no known information to guide rule selection. It would be desirable to be able to delay the use of the splitting rules until the search determines them to be useful. That is, backchain until a goal is obtained that matches the head of a non-Horn clause, and
then use the appropriate splitting rule. Plaisted suggests the delayed use of splitting rules in [8] by introducing a double-sequent notation that allows for a "bidirectional" generation of Gentzen-style refutations. When applied to SPRF, this notation implies a search utilizing the splitting rules in a delayed manner. Perhaps a clearer way to accomplish this same result is to alter the form of the splitting rule itself.

Instead of forcing the splitting rule to have \textit{FALSE} as goal in the lower sequent, we will allow the goal to be any head of the non-Horn clause. The upper sequents remain as before except that sequent whose assumption is the goal of the lower sequent is not included. For a non-Horn clause \textquote{\(H_1; \ldots; H_m : \leftarrow L_1, \ldots, L_n\)}, we then have \(m\) inference rules of the following form:

\[
\frac{\Gamma \rightarrow L_1 \quad \cdots \quad \Gamma \rightarrow L_n \quad \Gamma, H_1 \rightarrow \text{FALSE} \quad \cdots 
\quad \Gamma, H_m \rightarrow \text{FALSE}}{\Gamma \rightarrow H_k}
\]

for \(k = 1, \ldots, m\), where the notation \(\ldots \) represents the fact that \(\Gamma, H_k \rightarrow \text{FALSE}\) is omitted from the sequence. We will call these new rules \textit{delayed splitting rules}, since they can be used to implement the splitting rules of SPRF in a delayed manner. (The reader may note that the branches of the delayed splitting rule have corresponding branches in SPRF's splitting rule. In a backchaining search, the steps below the delayed splitting rule which generated the sequent \(\Gamma \rightarrow H_k\) correspond to the branch of the splitting rule with assumption \(H_k\). See the example below for an illustration of this.) We will call the variant of SPRF which uses these delayed splitting rules \textit{SPRF with Delay} (SPRF-D).

\textit{Example 3.2.} Consider our example clauses and their corresponding SPRF-D rules (note that the Horn rules are exactly as in SPRF):

\[
\begin{align*}
\text{FALSE := } h_1, \ y. & \quad \frac{\Gamma \rightarrow h_1 \quad \Gamma \rightarrow y}{\Gamma \rightarrow \text{FALSE}} \quad H1 \\
\text{FALSE := } h_2. & \quad \frac{\Gamma \rightarrow \text{FALSE}}{\Gamma \rightarrow h_2} \quad H2 \\
\text{y.} & \quad \frac{\Gamma \rightarrow y}{\Gamma \rightarrow h_1} \quad DS1 \\
\text{h}_1; \text{h}_2 := y. & \quad \frac{\Gamma \rightarrow y \quad \Gamma, h_2 \rightarrow \text{FALSE}}{\Gamma \rightarrow h_1} \quad DS1 \quad \frac{\Gamma \rightarrow y \quad \Gamma, h_1 \rightarrow \text{FALSE}}{\Gamma \rightarrow h_2} \quad DS2
\end{align*}
\]

These inference rules, along with the axioms, generate the following refutation:

\[
\frac{\frac{h_2 \rightarrow \text{FALSE}}{\frac{h_2 \rightarrow h_2 \quad H2}{\frac{\rightarrow h_1}{\frac{\rightarrow y \quad H3}{\frac{\rightarrow y \quad H3}{\frac{\rightarrow y \quad H3}}{\rightarrow h_1} \quad DS2}}{\rightarrow h_1} \quad DS1}}{\rightarrow \text{FALSE}}}{\rightarrow y \quad H1}
\]

If we compare this SPRF-D refutation with the SPRF refutation in Example 3.1, we see that the steps above the delayed splitting rule correspond to the steps in the leftmost and rightmost branches of the splitting rule. Similarly, the steps below the
A COMPARISON OF THREE PROLOG EXTENSIONS

delayed splitting rule correspond to the steps in the middle branch of the splitting rule (except that the assumption \( h_1 \) is implicit in SPRF-D and explicit in SPRF).

While the added control of SPRF-D is a big advantage, it should be noted that there are some disadvantages. First of all, SPRF-D requires multiple rules for each non-Horn clause, whereas SPRF only requires one rule per clause (however, the increase in the number of rules is bounded by the number of non-Horn clause heads). Another disadvantage is that the delayed use of splitting rules can lead to duplicated steps and hence longer proofs. This is due to the fact that steps below a delayed splitting rule do not have the explicit assumption found in the corresponding branch of the SPRF splitting rule. For example, if we altered the above clause set slightly by replacing the clause \( FALSE :\neg h_1, y. \) with \( FALSE :\neg h_1, w. \) and adding \( w :\neg h_1. \), we would find that a SPRF-D refutation of the clauses would require two duplicate delayed splitting rules (one for each occurrence of the sequent \( \rightarrow h_1 \)), whereas the SPRF refutation would require only one splitting rule. Similarly, steps can be duplicated when more than one delayed splitting rule is used.

The reader may note the similarity between the SPRF-D refutation above and the InH-PROLOG refutation presented in Example 2.1. Indeed, if we place the two refutations side by side, we may note a mapping between the lines (ignoring lines with empty continuations) and sequents. In the SPRF-D refutation below, the sequents are numbered to match with the corresponding lines in the InH-PROLOG refutation.

\[
\begin{align*}
1) & \quad FALSE \\
2) & \quad h_1, y \\
3) & \quad y, y \quad \# \quad \{h_2\} \\
4) & \quad y \quad \# \quad \{h_2\} \\
5) & \quad \# \quad \{h_2\} \\
6) & \quad FALSE \quad \# \quad h_2 \\
7) & \quad h_2 \rightarrow h_2 \\
8) & \quad \# \quad h_2 \\
\end{align*}
\]

This example highlights the fact that these systems behave almost identically. The SPRF-D axiom \( \Gamma \rightarrow L \) for \( L \in \Gamma \) corresponds to cancellation in InH-PROLOG, where an active head (assumption) satisfies a goal. For example, sequent 7 in the SPRF-D refutation above is an instance of the SPRF-D axiom, while the corresponding line in the InH-PROLOG refutation produces a cancellation. SPRF-D’s Horn rules correspond to InH-PROLOG’s PROLOG-like reduction via Horn clauses. In the SPRF-D refutation above, the Horn rule \( H1 \) generates subgoals \( h_1 \) and \( y \) from the initial goal \( FALSE \), just as reduction produces these same subgoals at the start of the InH-PROLOG refutation. Finally, SPRF-D’s delayed splitting rules correspond to InH-PROLOG’s reduction via non-Horn clauses, where the refutation is divided into cases. The delayed splitting rule yields one case which ignores the auxiliary heads of the non-Horn clause and other cases which assume these heads as conditional facts. Recall that in InH-PROLOG, these same cases are represented by blocks. In the examples above, sequents 1 through 4 of the SPRF-D refutation correspond to the start block of the
InH-PROLOG refutation (representing the case $P \cup \{h_1 : \neg y.\}$), and sequents 6 and 7 correspond to the restart block (representing the case $P \cup \{h_2.\}$).

In general, SPRF-D and InH-PROLOG differ in only two respects: (1) InH-PROLOG imposes a specific ordering on reduction, namely that a block must be completed before computation on any other block can begin; (2) InH-PROLOG has the cancellation pruning rule, which is not found in SPRF-D. These two differences are linked together. The value of the imposed ordering is due to the observation that if a block does not contain a cancellation by the distinguished active head, then the case analysis that introduced that block was unnecessary. Thus, if a block concludes without such a cancellation having taken place, the search may fail or backtrack. InH-PROLOG requires the entire block to be completed first, so an unsuccessful block (one without a cancellation by the distinguished active head) can be recognized before any computation is wasted on other blocks. This cancellation pruning rule reduces the search space considerably, as well as shortening refutations by disallowing unnecessary case analysis. An equivalent rule could easily be added to SPRF-D by requiring that every subtree with root sequent of the form $'A_1, \ldots, A_j \rightarrow \text{FALSE}'$ must contain the axiom $'A_1, \ldots, A_j, A_j'$. However, the gain is maximized when the search order is also altered as in InH-PROLOG. In SPRF-D, this amounts to reducing all sequents with a given assumption set before any others. Hence, we may view InH-PROLOG as identical to the version of SPRF-D which fully takes advantage of this pruning rule.

The relationship described here is stated formally in the following theorem:

**Theorem 1.** There exists a bijective mapping between InH-PROLOG refutations and SPRF-D refutations in which the cancellation pruning rule is enforced (i.e., every subtree with root sequent of the form $'A_1, \ldots, A_j \rightarrow \text{FALSE}'$ contains the axiom $'A_1, \ldots, A_j, A_j'$ as a leaf).

**Proof.** See Appendix A. $\square$

### 3.2. Further Comparisons

We have seen that InH-PROLOG and SPRF-D behave identically except that InH-PROLOG orders the search so as to make best use of a powerful pruning rule. To compare InH-PROLOG directly with SPRF, we need to return to the differences between SPRF and SPRF-D. SPRF requires only one rule for each non-Horn clause, whereas InH-PROLOG implicitly requires multiple rules (each non-Horn clause must be accessible from each of its heads). SPRF refutations also may be shorter, since InH-PROLOG may have to duplicate steps due to its delayed use of case analysis. On the other hand, InH-PROLOG’s delayed use of case analysis is a powerful control feature, as is the cancellation pruning rule. In addition, the cancellation pruning rule can lead to shorter refutations, since it disallows unnecessary case analysis.

We noted earlier that in Plaisted’s description of SPRF, splitting rules are not restricted to the literal $\text{FALSE}$ as goal in the lower sequent, even though this generality is not required for completeness. The flexibility in choosing the goal allows splitting rules to occur at any point in the refutation instead of just at the bottom. This same idea of performing the case analysis at any point in the search is
found in Loveland's original nH-PROLOG, which keeps an ancestor list and allows restart blocks to begin with any goal from this list.

The variant of SPRF currently implemented by Plaisted is referred to as the *modified SPRF*. Plaisted developed the modified SPRF in [8] by introducing a new predicate to represent negation. Each non-Horn clause is rewritten with all but one of the heads moved to the body and negated. For example, the clause ‘\(H_1 \land H_2 \leftarrow B\)’ would be written ‘\(H_1 \leftarrow B, \text{not}(H_2)\)’ or ‘\(H_2 \leftarrow B, \text{not}(H_1)\)’, but not both. The transformed non-Horn clauses are then treated just like Horn clauses. The only remaining non-Horn clause is ‘\(L \leftarrow \text{not}(L)\)’, which defines the predicate for negation and generates the only splitting rule. The modified SPRF then utilizes the double-sequent notation to allow for the delayed use of the lone splitting rule. Much of the similarity between InH-PROLOG and this variant of SPRF holds in the manner shown above, but a new tradeoff is introduced. The modified SPRF effectively orders the heads of a non-Horn clause so that only one head is accessible. This further restricts the search space. However, the ordering of the non-Horn clauses can be shown to be incompatible with the cancellation pruning rule of InH-PROLOG. In fact, examples exist which show that InH-PROLOG with ordered non-Horn clauses is incomplete, and modified SPRF with the pruning rule is incomplete.

4. N-PROLOG

The third system we will consider is *N-PROLOG*, developed by Gabbay and Reyle [3, 21. The main design goal of N-PROLOG is to extend PROLOG to handle hypothetical implications, allowing for arbitrary nestings of implications. It is essentially an intuitionistic system, shown to be complete and sound for positive intuitionistic logic (intuitionistic logic without negation). Gabbay also shows that it can be extended to positive classical logic with the addition of another rule (we discuss this extension later). However, if the input is restricted to be of certain forms and disjunction is classically defined, N-PROLOG becomes complete and sound for the full classical logic without altering its rules. Since we will focus on N-PROLOG with such restricted input forms, we stress the fact that our description does not present a complete picture of the system and its merits. However, it is interesting to see that both InH-PROLOG and SPRF can be embedded into N-PROLOG.

Sequents in N-PROLOG are of the form ‘\(P \vdash F\)’ where \(F\) is a formula and \(P\) is a set of formulas. The set \(P\) represents the input set plus any assumptions added to the set, and the formula \(F\) represents the query formula. Intuitively, a successful derivation of the sequent ‘\(P \vdash F\)’ shows that the query formula \(F\) follows (intuitionistically, in the primary variant) from \(P\). Formulas are written using only the connectives for conjunction (\(\land\)) and implication (\(\rightarrow\)). As with the previous systems, we will concern ourselves only with refutations. Thus, we will show a set of input formulas \(P_N\) to be unsatisfiable by showing that the sequent ‘\(P_N \vdash \text{FALSE}\)’ succeeds (i.e. by deriving it). N-PROLOG has the following rules:

**Conjunction rule:** ‘\(P \vdash (A_1 \land \cdots \land A_n)\)’ succeeds iff ‘\(P \vdash A_i\)’ succeeds for all \(i\).

**Implication rule:** ‘\(P \vdash (A \rightarrow B)\)’ succeeds iff ‘\(P + A \vdash B\)’ succeeds, where ‘\(P + A\)’ is to be understood as adding each conjunct of \(A\) separately to \(P\).
Atomic rules: For $q$ atomic, ‘$P ? q$’ succeeds iff either

1. $q \in P$, or
2. for some clause ‘$(C_1 \land \cdots \land C_n) \rightarrow q$’ in $P$ we have that ‘$P ? (C_1 \land \cdots \land C_n)$’ succeeds.

Since our goal is to compare N-PROLOG with the previously presented systems, we will restrict the input set to be in clausal form. Unlike InH-PROLOG and SPRF, however, N-PROLOG is $\{ \land, \rightarrow \}$ based. Since Horn clauses are already written using only conjunction and implication, there is no problem in representing Horn clauses: the clause ‘$H : \leftarrow L_1, \ldots, L_n$’ is represented by ‘$(L_1 \land \cdots \land L_n) \rightarrow H$’. Again, we assume a new literal FALSE has been added so that all-negative clauses appear as Horn clauses with head FALSE. However, non-Horn clauses are not expressible in the intuitionistic N-PROLOG. We must add a classical definition of disjunction in order to represent these clauses. Using two different representations (classically equivalent but not intuitionistically equivalent), we will show how SPRF and InH-PROLOG can be embedded in N-PROLOG.

4.1. Embedding SPRF in N-PROLOG

The first representation of non-Horn clauses is suggested by Gabbay in [2]. We start with the classical definition of disjunction

$$H_1 \lor H_2 \equiv \neg (\neg(H_1) \land \neg(H_2))$$

$$\equiv ((H_1 \rightarrow \text{FALSE}) \land (H_2 \rightarrow \text{FALSE})) \rightarrow \text{FALSE}. $$

Extending this, a non-Horn clause ‘$H_1; \ldots; H_m : \leftarrow L_1, \ldots, L_n$’ which represents the formula ‘$L_1 \land \cdots \land L_n \rightarrow H_1 \lor \cdots \lor H_m$’ is written

$$L_1 \land \cdots \land L_n \rightarrow \left[ ((H_1 \rightarrow \text{FALSE}) \land \cdots \land (H_m \rightarrow \text{FALSE})) \rightarrow \text{FALSE} \right].$$

We may note that this formula is equivalent (even in intuitionistic logic) to

$$[ L_1 \land \cdots \land L_n \land (H_1 \rightarrow \text{FALSE}) \land \cdots \land (H_m \rightarrow \text{FALSE}) ] \rightarrow \text{FALSE}.$$

Thus, we may invoke this implicit lemma and represent a non-Horn clause in this last form. (The reader may recognize the structural similarity between this representation and the corresponding splitting rule of SPRF.) With non-Horn clauses represented, let us return to our running example:

Example 4.1. The clause set is translated into N-PROLOG notation:

$$\text{FALSE} :\leftarrow h_1, y. \quad \Rightarrow \quad (h_1 \land y) \rightarrow \text{FALSE}$$

$$\text{FALSE} :\leftarrow h_2. \quad \Rightarrow \quad h_2 \rightarrow \text{FALSE}$$

$$y. \quad \Rightarrow \quad y$$

$$h_1; h_2 :\leftarrow y. \quad \Rightarrow \quad [ y \land (h_1 \rightarrow \text{FALSE}) \land (h_2 \rightarrow \text{FALSE}) ] \rightarrow \text{FALSE}$$
Referring to this set of formulas as $P_N$, we have the following N-PROLOG refutation (the rule used to derive each line is listed to the right):

1) $P_N \rightarrow FALSE$
2) $P_N \rightarrow [y \land (h_1 \rightarrow FALSE) \land (h_2 \rightarrow FALSE)]$ Atomic rule (2)
3) $P_N \rightarrow y, P_N \rightarrow (h_1 \rightarrow FALSE), P_N \rightarrow (h_2 \rightarrow FALSE)$ Conj. rule
4) $P_N \rightarrow (h_1 \rightarrow FALSE), P_N \rightarrow (h_2 \rightarrow FALSE)$ Atomic rule (1)
5) $P_N \rightarrow h_1 ? FALSE, P_N \rightarrow (h_2 \rightarrow FALSE)$ Impl. rule
6) $P_N \rightarrow h_1 ? (h_1 \land y), P_N \rightarrow (h_2 \rightarrow FALSE)$ Atomic rule (2)
7) $P_N \rightarrow h_1 ? h_1, P_N \rightarrow h_1 ? y, P_N \rightarrow (h_2 \rightarrow FALSE)$ Conj. rule
8) $P_N \rightarrow h_1 ? y, P_N \rightarrow (h_2 \rightarrow FALSE)$ Atomic rule (1)
9) $P_N \rightarrow (h_2 \rightarrow FALSE)$ Atomic rule (1)
10) $P_N \rightarrow h_2 ? FALSE$ Impl. rule
11) $P_N \rightarrow h_2 ? h_2$ Atomic rule (2)

Since we are restricting the form of the input formulas, we can combine the N-PROLOG rules to obtain a simpler, condensed description. First, since the initial goal is $FALSE$ and nested conjunctions are not allowed, the only conjunctive goal we will encounter will be introduced by the second atomic rule. Thus, instead of having the extra step of writing a sequent with a conjunctive goal and then applying the conjunction rule, we can condense these steps by having the atomic rule produce the end result directly. Similarly, we may note that the only time we will encounter a goal with an implication is as a result of using the second atomic rule with a non-Horn formula. We may again condense steps by having the atomic rule produce directly the result of the implication rule. Hence, we obtain the following condensed set of rules:

**Atomic rules:** For $q$ atomic, `$P \rightarrow q$' succeeds iff any of the following holds:

1. $q \in P$.

2. $q$ is the literal $FALSE$ and for some non-Horn formula
   
   $[[L_1 \land \cdots \land L_n \land (H_1 \rightarrow FALSE) \land \cdots \land (H_m \rightarrow FALSE)] \rightarrow FALSE$ 

   in $P$ we have that `$P \rightarrow L_i$' succeeds for all $i$ and `$P + H_j \rightarrow FALSE$' succeeds for all $j$.

3. For some Horn formula `(L_1 \land \cdots \land L_n) \rightarrow q` in $P$ we have that `$P \rightarrow L_i$' succeeds for all $i$.

Using these condensed rules, we present a refutation of our example clauses below. Note that it is identical to the refutation above except that the sequents containing conjunctions and implications are removed:

1) $P_N \rightarrow FALSE$
2) $P_N \rightarrow y, P_N \rightarrow h_1 ? FALSE, P_N \rightarrow h_2 ? FALSE$ Rule (2)
3) $P_N \rightarrow h_1 ? FALSE, P_N \rightarrow h_2 ? FALSE$ Rule (1)
4) $P_N \rightarrow h_1 ? h_1, P_N \rightarrow h_1 ? y, P_N \rightarrow h_2 ? FALSE$ Rule (3)
5) $P_N \rightarrow h_1 ? y, P_N \rightarrow h_2 ? FALSE$ Rule (1)
6) $P_N \rightarrow h_2 ? FALSE$ Rule (1)
7) $P_N \rightarrow h_2 ? h_2$ Rule (3)

Thus, we obtain the following condensed refutation:

1) $P_N \rightarrow FALSE$
2) $P_N \rightarrow y, P_N \rightarrow h_1 ? FALSE, P_N \rightarrow h_2 ? FALSE$ Rule (2)
3) $P_N \rightarrow h_1 ? FALSE, P_N \rightarrow h_2 ? FALSE$ Rule (1)
4) $P_N \rightarrow h_1 ? h_1, P_N \rightarrow h_1 ? y, P_N \rightarrow h_2 ? FALSE$ Rule (3)
5) $P_N \rightarrow h_1 ? y, P_N \rightarrow h_2 ? FALSE$ Rule (1)
6) $P_N \rightarrow h_2 ? FALSE$ Rule (1)
7) $P_N \rightarrow h_2 ? h_2$ Rule (3)
The reader may note the similarity between this refutation and the SPRF refutation presented in Example 3.1. In fact, there is a mapping between the lines of the condensed N-PROLOG refutation and the sequents of the SPRF refutation. We list below the SPRF refutation with the sequents numbered to match the N-PROLOG lines:

\[
\begin{array}{cccc}
3) & h_1 \rightarrow \text{FALSE} & 5) & h_1 \rightarrow y^{H^3} \\
4) & h_1 \rightarrow h_1 & 6) & h_2 \rightarrow \text{FALSE}^{H^2} \\
2) & y^{H^3} & 7) & h_2 \rightarrow h_2 \\
1) & \rightarrow \text{FALSE} & & s_1.
\end{array}
\]

In general, we can see that this condensed version of N-PROLOG and SPRF behave identically. Rule (1) corresponds to the SPRF axioms, where a goal is trivially proven if it is an assumption or fact. For example, lines 2, 4, 5, and 7 in the condensed N-PROLOG refutation above have rule (1) applied to them, while the corresponding sequents in the SPRF refutation are instances of axioms (sequents 2 and 5 are instances of the degenerate Horn rule $H^3$, which we view as an axiom). Rule (2) corresponds to the splitting rules of SPRF, where the goal $\text{FALSE}$ is proven by proving each goal in the body of the clause as well as showing that $\text{FALSE}$ follows from each of the heads. In the condensed N-PROLOG refutation above, rule (2) is applied to the first line, just as splitting rule $s_1$ is applied to the initial sequent of the SPRF refutation. Finally, rule (3) corresponds to the Horn inference rules of SPRF, where the head is proven by proving each of the goals from the body of the clause. In the above examples, line 7 is obtained from line 6 in the condensed N-PROLOG refutation via rule (3), while the corresponding sequent in the SPRF refutation is obtained via Horn rule $H^2$. Noting the identical behavior of the two systems, we see that restricting the input to clausal form and choosing this representation for non-Horn clauses allow us to embed SPRF into N-PROLOG.

The relationship described here is stated formally in the following theorem:

**Theorem 2.** There exists a bijective mapping between condensed N-PROLOG refutations (as described above) and SPRF refutations.

**Proof.** See Appendix B. $_\square$

### 4.2. Embedding InH-PROLOG in N-PROLOG

In comparing N-PROLOG with SPRF, we chose to represent the non-Horn clause 

\[ H_1; \ldots; H_m \leftarrow L_1, \ldots, L_n \]

as

\[ [L_1 \land \cdots \land L_n \land (H_1 \rightarrow \text{FALSE}) \land \cdots \land (H_m \rightarrow \text{FALSE})] \rightarrow \text{FALSE}. \]

We may also note that this formula is logically equivalent (in classical logic, not intuitionistic logic) to the formula

\[ [L_1 \land \cdots \land L_n \land (H_2 \rightarrow \text{FALSE}) \land \cdots \land (H_m \rightarrow \text{FALSE})] \rightarrow H_1. \]

Likewise, we could move any one of the $H_i$'s to the conclusion of the implication. For this comparison, we will represent each non-Horn clause by all $m$ formulas of
the form

\[ \left[ L_1 \land \cdots \land L_n \land (H_1 \rightarrow FALSE) \land \cdots \right. \]

\[ \land (H_{k-1} \rightarrow FALSE) \land (H_{k+1} \rightarrow FALSE) \land \cdots \land (H_m \rightarrow FALSE) \left. \right] \rightarrow H_k \]

for \( k = 1, \ldots, m \). (Again, the reader may recognize the structural similarity between these representations and the delayed splitting rules of SPRF-D.) We may note that these \( m \) formulas are implicitly ANDed together, and their conjunction is classically equivalent to the original formula. With this representation of non-Horn clauses, we return to our example:

**Example 4.2.** The set of formulas is translated into N-PROLOG notation (note only the non-Horn clause is represented differently):

\[
\begin{align*}
\text{FALSE} & \leftarrow h_1, y. \quad \Rightarrow \quad (h_1 \land y) \rightarrow \text{FALSE} \\
\text{FALSE} & \leftarrow h_2. \quad \Rightarrow \quad h_2 \rightarrow \text{FALSE} \\
y. \quad \Rightarrow \quad y \\
h_1; h_2 & \leftarrow y. \quad \Rightarrow \quad [y \land (h_2 \rightarrow \text{FALSE})] \rightarrow h_1 \\
& \quad \Rightarrow \quad [y \land (h_1 \rightarrow \text{FALSE})] \rightarrow h_2
\end{align*}
\]

Referring to this set of formulas as \( P_{N_2} \), we have the following N-PROLOG refutation:

1) \( P_{N_2} \leftarrow \text{FALSE} \)
2) \( P_{N_2} \leftarrow (h_1 \land y) \) \quad Atomic rule (2)
3) \( P_{N_2} \leftarrow h_1, P_{N_2} \leftarrow y \) \quad Conj. rule
4) \( P_{N_2} \leftarrow [y \land (h_2 \rightarrow \text{FALSE})], P_{N_2} \leftarrow y \) \quad Atomic rule (2)
5) \( P_{N_2} \leftarrow y, P_{N_2} \leftarrow (h_2 \rightarrow \text{FALSE}), P_{N_2} \leftarrow y \) \quad Conj. rule
6) \( P_{N_2} \leftarrow (h_2 \rightarrow \text{FALSE}), P_{N_2} \leftarrow y \) \quad Atomic rule (1)
7) \( P_{N_2} \leftarrow h_2 \leftarrow \text{FALSE}, P_{N_2} \leftarrow y \) \quad Impl. rule
8) \( P_{N_2} \leftarrow h_2 \leftarrow h_2, P_{N_2} \leftarrow y \) \quad Atomic rule (2)
9) \( P_{N_2} \leftarrow y \) \quad Atomic rule (1)
\( \square \) \quad Atomic rule (1)

As in the previous section, we can combine some of the N-PROLOG rules due to the restricted input form. The conjunction rule and the implication rule can be incorporated into the atomic rule to obtain the following condensed set of rules:

**Atomic rules.** For \( q \) atomic we have \( 'P \leftarrow q' \) succeeds iff either

1) \( q \in P \), or
2) for some clause \( ['L_1 \land \cdots \land L_n \land (H_1 \rightarrow \text{FALSE}) \land \cdots \land (H_m \rightarrow \text{FALSE})] \rightarrow q' \) in \( P \) we have that \( 'P \leftarrow L_i' \) succeeds for all \( i \) and \( 'P + H_j \leftarrow \text{FALSE}' \) succeeds for all \( j \).
As before, we list below a refutation of the example clauses using the condensed rules:

1) \( P_{N_2} \not\rightarrow \text{FALSE} \)
2) \( P_{N_2} \not\rightarrow h_1, P_{N_2} \not\rightarrow y \quad \text{Rule (2)} \)
3) \( P_{N_2} \not\rightarrow y, P_{N_2} + h_2 \not\rightarrow \text{FALSE}, P_{N_2} \not\rightarrow y \quad \text{Rule (2)} \)
4) \( P_{N_2} + h_2 \not\rightarrow \text{FALSE}, P_{N_2} \not\rightarrow y \quad \text{Rule (1)} \)
5) \( P_{N_2} + h_2 \not\rightarrow h_2, P_{N_2} \not\rightarrow y \quad \text{Rule (2)} \)
6) \( P_{N_2} \not\rightarrow y \quad \text{Rule (1)} \)

The reader may note the similarity between this refutation and the SPRF-D refutation presented in Example 3.2. As before, there is a mapping between the lines of the condensed N-PROLOG refutation and the sequents of the SPRF-D refutation. We list below the SPRF-D refutation with the sequents numbered to match the N-PROLOG lines:

\[
\begin{align*}
5) & \quad h_2 \rightarrow h_2 \\
3) & \quad \rightarrow y^H3 \\
4) & \quad h_2 \rightarrow \text{FALSE}^H2 \\
2) & \quad \rightarrow h_1^{DS1} \\
6) & \quad \rightarrow y^H3 \\
1) & \quad \rightarrow \text{FALSE}^{H1}.
\end{align*}
\]

In general, we can see that this condensed version of N-PROLOG and SPRF-D behave identically. Rule (1) corresponds to the SPRF-D axioms, where a goal is trivially proven if it is an assumption or fact. For example, lines 3, 5, and 6 in the condensed N-PROLOG refutation above have rule (1) applied to them, while the corresponding sequents in the SPRF refutation are instances of axioms (sequent 3 and 6 are instances of the degenerate Horn rule \( H^3 \), which we view as an axiom). Rule (2) corresponds to all of the inference rules of SPRF-D, both the Horn rules and delayed splitting rules, where the head is proven by proving each of the goals from the body of the clause as well as showing that \( \text{FALSE} \) follows from each of the heads (in the case where the clause is non-Horn). In the above examples, lines 2 and 4 in the condensed N-PROLOG refutation both have rule (2) applied to them, while sequent 2 in the SPRF refutation has delayed splitting rule \( DS1 \) applied to it and sequent 4 has Horn rule \( H^2 \) applied to it. Noting the identical behavior of the two systems, we see that restricting the input to clausal form and choosing this representation for non-Horn clauses allow us to embed SPRF-D into N-PROLOG. Since we have already shown that InH-PROLOG is identical to the version of SPRF-D which utilizes the cancellation pruning rule, the embedding follows for InH-PROLOG.

The relationships described here are stated formally in the following theorems:

**Theorem 3.** There exists a bijective mapping between condensed N-PROLOG refutations (as described above) and SPRF-D refutations.

**PROOF.** See Appendix C. \( \square \)

**Theorem 4.** There exists a bijective mapping between condensed N-PROLOG refutations in which the cancellation pruning rule is enforced (i.e., every assumed literal is used in a rule (1) application) and InH-PROLOG refutations.

**PROOF.** See Appendix C. \( \square \)
4.3. Further Comparisons

As noted earlier, in [2] Gabbay presents a variant of N-PROLOG which is complete and sound for positive classical logic. This variant, called NR-PROLOG, is simply N-PROLOG with an additional rule called the restart rule. While we will not present this variant, it is interesting to note that the addition of the restart rule allows us to relate InH-PROLOG and NR-PROLOG by use of a different definition of disjunction, namely

\[ H_1 \lor H_2 \equiv (H_2 \rightarrow H_1) \rightarrow H_1. \]

The relationship between InH-PROLOG and NR-PROLOG is not quite as direct as were previous mappings; it is possible on occasion to obtain redundant computation in NR-PROLOG not present in InH-PROLOG. As the names suggest, the restart rule in NR-PROLOG and a restart line in InH-PROLOG are linked in that the restart rule produces a sequent that corresponds to a restart line.

5. CONCLUSION

Some variant of each of the three systems described here has been implemented by its author and/or colleagues: nH-PROLOG and N-PROLOG as programming languages that extend PROLOG, and SPRF as a general-purpose theorem prover. We have demonstrated that although these three systems were developed simultaneously from very different approaches, they are nonetheless closely related. This suggests that their underlying reasoning methods, in particular the case analysis which performs non-Horn reasoning, are general and intuitive.

While they are similar, the systems do have essential differences which can make one of the systems more attractive for a given domain. We have already seen that the major tradeoff between InH-PROLOG and SPRF is that InH-PROLOG better controls the search while SPRF has a smaller set of rules. Similarly, when comparing InH-PROLOG with the currently implemented variant of SPRF, modified SPRF, it can be seen that InH-PROLOG allows for the cancellation pruning rule while modified SPRF allows for the ordering of clause heads. Each of these features prunes the search space, so depending upon which type of pruning was more useful in a given domain, the appropriate system could be chosen. For example, the cancellation pruning rule appears to be the more effective of the two when using a PROLOG-like depth-first search of the clauses, making InH-PROLOG the more attractive choice as a programming language. In a theorem proving environment with a complete search strategy, however, the smaller rule set of modified SPRF might be more advantageous. Similarly, N-PROLOG might be the system of choice in another domain because of its more general input form. However, since N-PROLOG is essentially an intuitionistic system, classical completeness can only be obtained by restricting the input to certain forms (e.g. the forms presented here) or else extending the system.

We make a final remark that the mappings established here between InH-PROLOG and the alternative systems, SPRF and N-PROLOG, do not hold for the original nH-PROLOG of [4, 5], although strong similarities exist, of course. The original version of nH-PROLOG lacks the inheritance of active heads, so each block has only one active head. Thus, cancellation is a constant-time operation.
which guarantees a fast inner-loop speed (although coding techniques can mitigate this disadvantage for InH-PROLOG in compilers [6]). However, the restart rule is more complex, and completeness is much more difficult to establish than for InH-PROLOG.

APPENDIX A

We now prove Theorem 1, which formalizes the relationship between InH-PROLOG and SPRF-D. First, we may recognize that while InH-PROLOG was designed as a programming language and hence presented in a linearized format, the rules lend themselves to a proof-tree notation as well. We can rewrite the InH-PROLOG rules of Section 2 as follows:

**Rule 1.** Axiom: ‘G#A,, A,’ where G matches with some active head Aj.

Note that this axiom corresponds to cancellation [step (2) (a)] in the linear description found in Section 2, where an active head satisfies a goal.

**Rule 2.** For each clause ‘H,, Hm :- L,, Ln,’ there are m inference rules of the form

\[
\frac{L_1#A_1, \ldots, A_i \cdots L_n#A_1, \ldots, A_i (H_1)A_i, \ldots, A_i \cdots (H_m)A_i, \ldots, A_i}{H_k#A_1, \ldots, A_i}.
\]

where a sequent of the form ‘{(H)_A,, A,’ represents the deferred head H (the active heads are written as subscripts for notational convenience, so that the information needed to perform a restart is localized). The notation ‘\( \cdots \# \cdots \)’ here represents the fact that ‘{(H)_A,, A,’ is omitted from the sequence. Note that this rule corresponds to reduction [step (2) (b)] in the linear description of Section 2, where the goal is replaced by the literals in the body of the called clause and each auxiliary head (if any) is deferred. As in SPRF and SPRF-D, a fact can be viewed as a degenerate Horn clause which produces a rule with no upper sequents, i.e. an axiom.

**Rule 3.** For each deferred head H there is the inference rule:

\[
\frac{FALSE#H, A_i, \ldots, A_i}{(H)_A, \ldots, A_i}.
\]

Note that this rule corresponds to the restart rule [step (3)] in the linear description of Section 2, where a deferred head H produces a restart line with goal FALSE and active heads H, A,, A (assuming A,, A were the active heads in the block where H was deferred).

We may also note that step (1) of the linear description states that the first line of a refutation must be of the form ‘FALSE’, so the root sequent of a refutation in tree form must also be the sequent ‘FALSE’. Step (4) of the linear description asserts the cancellation pruning rule, which can be stated here as requiring that every subtree with root sequent of the form ‘FALSE#A,, A,’ must contain the axiom ‘A,#A,, A,’ as a leaf.
A COMPARISON OF THREE PROLOG EXTENSIONS

Written in the above notation, an InH-PROLOG refutation is a derivation of the sequent ‘FALSE’ from axioms (instances of rule 1 and degenerate forms of rule 2) via the generated inference rules (instances of rule 2 and rule 3). Furthermore, the cancellation pruning rule as described above is enforced.

For the following proofs, we define a deduction (in any system) to be a derivation of a sequent from axioms via the inference rules of that system. (Note that a refutation is a deduction which has a special root sequent.) We define the transformation $T_1$ on deductions as follows: $T_1$ removes every sequent of the form \( {'(H)_{A_1}, \ldots, A_i'} \) and replaces each sequent of the form \( {'A_1, \ldots, A_j \rightarrow G'} \) with the sequent \( {'A_1, \ldots, A_j \rightarrow G'} \). We say that $D^{T_1}$ is the deduction obtained by applying $T_1$ to deduction $D$. Likewise, $D^{T_1^{-1}}$ is the deduction obtained by applying the inverse transformation $T_1^{-1}$ to deduction $D$.

**Lemma 1.1.** Let $D$ be an InH-PROLOG deduction (as described above). Then $D^{T_1}$ is a valid SPRF-D deduction in which the cancellation pruning rule is enforced (i.e., every subtree with root sequent of the form ‘$A_1, \ldots, A_j \rightarrow FALSE$’ contains the axiom ‘$A_1, \ldots, A_j \rightarrow A_j$’ as a leaf).

**Proof** (By induction). We define the depth of a sequent in a deduction as follows: the depth of the root sequent is 1; if the lower sequent of a rule has depth $i$, then the upper sequents have depth $i + 1$. The depth of a deduction is the greatest depth of any sequent in the deduction. We now prove the lemma by induction using the following induction predicate:

$P(i)$. If InH-PROLOG deduction $D$ has depth $i$, then $D^{T_1}$ is a valid SPRF-D deduction in which the cancellation pruning rule is enforced.

**Base:** $P(1)$. $D$ must be a tree with single sequent \( {'G#A_1, \ldots, A_i'} \) where $G$ is a fact in the input set (in which case the sequent is an instance of a degenerate rule 2, i.e. an axiom) or $G$ matches one of the active heads $A_i$ (in which case the sequent is an instance of the axiom, rule 1). When we convert this sequent, we obtain the sequent \( {'A_1, \ldots, A_j \rightarrow G'} \). If $G$ is a fact, then there is a degenerate SPRF-D Horn rule, i.e. axiom, of the form \( {'A_1, \ldots, A_j \rightarrow G'} \). If $G$ matches one of the $A_i$'s, then this sequent is an instance of the SPRF-D axiom. In either case, $D^{T_1}$, which consists of this single sequent, is a valid SPRF-D deduction. And since there are no sequents in $D^{T_1}$ of the form ‘$A_1, \ldots, A_j \rightarrow FALSE$’, the cancellation pruning rule is not violated.

**Induction hypothesis:** Assume $P(k)$ for all $k < i$.

**Step:** Assume $D$ has depth $i$. One of the following two cases must hold:

1. The rule found at the root of $D$ is an instance of rule 2 of the form

   \[
   \frac{L_1#A_1, \ldots, A_i \quad \ldots \quad L_n#A_j, \ldots, A_i \quad A_i[H_1]_{A_i, \ldots, A_i} \quad \ldots \quad H_m\#A_j, \ldots, A_i}{H_i\#A_j, \ldots, A_i}
   \]

   Note that this rule indicates that \( {'H_i; \ldots; H_m \vdash L_1, \ldots, L_n'} \) is a clause in the input set. If we convert the sequents in this rule (since each sequent of the form \( {'(H)_{A_i, \ldots, A_i}'} \) is discarded by the transformation, we instead convert
the sequent above, namely ‘FALSE#H, A_j, ..., A_i’), we obtain the rule

\[
\frac{A \rightarrow L_1 \cdots A \rightarrow L_n}{A \rightarrow H_k}
\]

where \( A = A_1, \ldots, A_i \), which we see is a valid SPRF-D rule (given that ‘\( H_1, \ldots, H_m \vdash L_1, \ldots, L_n \)’ is in the input set). The upper sequents of this instance of rule 2 are the roots of deductions with depth less than \( i \), so the induction hypothesis guarantees that the converted deductions (whose roots are the upper sequents of the delayed splitting rule) are valid SPRF-D deductions in which the cancellation pruning rule is enforced. Thus, \( D_{T_1} \) is a valid SPRF-D deduction, and all proper subtrees of \( D_{T_1} \) enforce the cancellation pruning rule. To ensure that \( D_{T_1} \) itself does not violate the cancellation pruning rule, we must consider the case where \( H_k \) is FALSE.

If \( H_k \) is FALSE, then the cancellation pruning rule in InH-PROLOG requires that the axiom ‘\( A_j \# A_j, \ldots, A_i \)’ occur as a leaf in \( D \). Since this sequent converts to the SPRF-D axiom ‘\( A_1, \ldots, A_j \rightarrow A_i \)’, the cancellation pruning rule is not violated in \( D_{T_1} \). Hence, the entire converted refutation \( D_{T_1} \) is a valid SPRF-D deduction in which the cancellation pruning rule is enforced, i.e., \( P(i) \) holds.

(2) The rule found at the root of \( D \) is an instance of rule 3 of the form

\[
\frac{FALSE#H, A_j, ..., A_i}{\{H\} A_1, ..., A_i}.
\]

Since sequents of the form ‘\( \{H\} A_j, ..., A_i \)’ are discarded by the transformation \( T_i \), applying \( T_i \) to \( D \) produces the same deduction as applying \( T_i \) to the subtree whose root is the upper sequent ‘\( FALSE#H, A_j, ..., A_i \)’. Since this deduction has depth less than \( i \), the induction hypothesis guarantees that converting it yields a valid SPRF-D deduction in which the cancellation pruning rule is enforced. Hence, \( D_{T_i} \) is a valid SPRF-D deduction in which the cancellation pruning rule is enforced, i.e., \( P(i) \) holds. \( \square \)

Lemma 1.2. Let \( D \) be a SPRF-D deduction in which the cancellation pruning rule is enforced (i.e., every subtree with root sequent of the form ‘\( A_j, ..., A_j \rightarrow FALSE \)’ contains the axiom ‘\( A_1, ..., A_j \rightarrow A_j \)’ as a leaf). Then \( D_{T_1} \) is a valid InH-PROLOG deduction (as described above).

Proof. The proof follows by induction as in Lemma 1.1, with the roles of the systems reversed. \( \square \)

Theorem 1. There exists a bijective mapping between InH-PROLOG refutations (as described above) and SPRF-D refutations in which the cancellation pruning rule is enforced (i.e., every subtree with root sequent of the form ‘\( A_1, ..., A_j \rightarrow FALSE \)’ contains the axiom ‘\( A_1, ..., A_j \rightarrow A_j \)’ as a leaf).

Proof. Let \( T_1 \) be the proof transformation described above, which removes sequents of the form ‘\( \{H\} A_j, ..., A_i \)’ and replaces each sequent of the form ‘\( G#A_j, ..., A_i \)’ with the sequent ‘\( A_1, ..., A_j \rightarrow G \)’. 
\( \Rightarrow \): Let \( R \) be an InH-PROLOG refutation. Since \( R \) is also an InH-PROLOG deduction, it follows from Lemma 1.1 that \( R^{T_1} \) is a valid SPRF-D deduction in which the cancellation pruning rule is enforced. And since the root of \( R \) must be the sequent \( 'FALSE' \) (by definition of an InH-PROLOG refutation) which converts to the sequent \( '→ FALSE' \), it follows that \( R^{T_1} \) is also a valid SPRF-D refutation in which the cancellation pruning rule is enforced.

\( \Leftarrow \): Similarly, let \( R \) be a SPRF-D refutation in which the cancellation pruning rule is enforced. Since \( R \) is also a SPRF-D deduction, it follows from Lemma 1.2 that \( R^{T_1} \) is a valid InH-PROLOG deduction. And since the root of \( R \) must be the sequent \( '→ FALSE' \) (by definition of a SPRF-D refutation) which converts to the sequent \( 'FALSE' \), it follows that \( R^{T_1} \) is also a valid InH-PROLOG refutation.

\( \square \)

APPENDIX B

We now prove Theorem 2, which formalizes the relationship between N-PROLOG and SPRF. Like InH PROLOG, N-PROLOG was designed as a programming language and hence presented in a linearized format. However, the rules lend themselves to a proof-tree notation as well. We can rewrite the condensed N-PROLOG rules of Section 4.1 as follows:

Rule 1. Axiom: \( 'P \ ? q' \) where \( q \in P \)

Rule 2. For each formula of the form \( '[L_1 \ ∧ \cdots \ ∧ L_n \ ∧ (H_1 \ → \ FALSE) \ ∧ \cdots \ ∧ (H_m \ → \ FALSE)] \ → \ FALSE' \) in \( P \) there is the inference rule

\[
\begin{array}{c}
P \ ? L_1 \\
\vdots \\
P \ ? L_n \\
P + H_1 \ ? FALSE \\
\vdots \\
P + H_m \ ? FALSE
\end{array}
\]

\[ P \ ? FALSE \]

Rule 3. For each formula of the form \( '(L_1 \ ∧ \cdots \ ∧ L_n) \ → q' \) in \( P \) there is the inference rule

\[
\begin{array}{c}
P \ ? L_1 \\
\vdots \\
P \ ? L_n
\end{array}
\]

\[ P \ ? q \]

Let \( P_C \) denote the input set written in clausal form (i.e. as in SPRF), and let \( P_{N_1} \) denote the input set written as described in Section 4.1 (where the clauses are represented using only conjunction and implication). A refutation in this system is a derivation of the sequent \( 'P_{N_1} \ ? FALSE' \) from axioms (instances of rule 1) via the generated inference rules (instances of rule 2 and rule 3).

As in Appendix A, we define a deduction (in any system) to be a derivation of a sequent from axioms via the inference rule of that system. We define transformation \( T_2 \) on deductions as follows: \( T_2 \) removes the initial set \( P_{N_1} \) from the antecedent of each sequent and replaces the symbol \( '?' \) with \( '→' \). We say that \( D^{T_2} \) is the deduction obtained by applying \( T_2 \) to the deduction \( D \). Likewise, \( D^{T_2 \ -1} \) is the deduction obtained by applying the inverse transformation \( T_2^{-1} \) to the deduction \( D \).

Lemma 2.1. Let \( D \) be a condensed N-PROLOG deduction (as described above). Then \( D^{T_2} \) is a valid SPRF deduction.


Proof (By induction). We define the depth of a sequent in a deduction as follows: the depth of the root sequent is 1; if the lower sequent of a rule has depth \( i \), then the upper sequents have depth \( i + 1 \). The depth of a deduction is the greatest depth of any sequent in the deduction. We now prove the lemma by induction using the following induction predicate:

\[ P(i). \text{ If the condensed N-PROLOG deduction } D \text{ has depth } i, \text{ then } D^{T2} \text{ is a valid SPRF deduction.} \]

**Base:** \( P(1) \). \( D \) must be a tree with single sequent ‘\( P_{N_i} + \Gamma \Rightarrow q \)’ where \( P_{N_i} \) is the initial set, \( \Gamma \) is a set of additional assumptions, and \( q \) is a fact in either \( P_{N_i} \) (in which case the sequent is an instance of a degenerate rule 3, i.e. an axiom) or \( \Gamma \) (in which case the sequent is an instance of the axiom, rule 1). When we convert this sequent, we obtain the sequent ‘\( \Gamma \Rightarrow q \)’. If \( q \) is a fact in the initial set \( P_{N_i} \), then it is also a fact in \( P_C \), so there is a degenerate SPRF Horn rule, i.e. axiom, of the form ‘\( \Gamma \Rightarrow q \)’. If \( q \) is an assumption in \( \Gamma \), then this sequent is an instance of the SPRF axiom. In either case, \( D^{T2} \), which consists of this single sequent, is a valid SPRF deduction.

**Induction hypothesis:** Assume \( P(k) \), for all \( k < i \).

**Step:** Assume \( D \) has depth \( i \). One of the following two cases must hold:

1. The rule found at the root of \( D \) is an instance of rule 2 of the form

\[
\frac{P'_{N_i} \Rightarrow L_1 \ldots \cdot P'_{N_i} \Rightarrow L_n \cdot P'_{N_i} + H_1 \Rightarrow FALSE \ldots \cdot P'_{N_i} + H_m \Rightarrow FALSE}{P'_{N_i} \Rightarrow FALSE}
\]

where \( P'_{N_i} \) is the initial set \( P_{N_i} \) with additional assumptions \( \Gamma \). Note that this rule indicates that ‘\( [L_1 \wedge \ldots \wedge L_n \wedge (H_1 \Rightarrow FALSE) \wedge \ldots \wedge (H_m \Rightarrow FALSE)] \Rightarrow FALSE \)’ is a formula in \( P_{N_i} \), and so ‘\( H_1, \ldots, H_m \vdash L_1, \ldots, L_n \)’ is a clause in \( P_C \). If we convert the sequents in this rule, we obtain the rule

\[
\frac{\Gamma \Rightarrow L_1 \ldots \cdot \Gamma \Rightarrow L_n \cdot \Gamma, H_1 \Rightarrow FALSE \ldots \cdot \Gamma, H_m \Rightarrow FALSE}{\Gamma \Rightarrow FALSE}
\]

which we see is a valid SPRF splitting rule (given that ‘\( H_1, \ldots, H_m \vdash L_1, \ldots, L_n \)’ is in \( P_C \)). The upper sequents of this instance of rule 2 are the roots of deductions with depth less than \( i \), so the induction hypothesis guarantees that the converted deductions (whose roots are the upper sequents of the splitting rule) are valid SPRF deductions. Hence, the entire converted refutation \( D^{T2} \) is a valid SPRF deduction, i.e., \( P(i) \) holds.

2. The rule found at the root of \( D \) is an instance of rule 3 of the form

\[
\frac{P_{N_i} + \Gamma \Rightarrow L_1 \ldots \cdot P_{N_i} + \Gamma \Rightarrow L_n}{P_{N_i} + \Gamma \Rightarrow q}
\]

where \( P_{N_i} \) is the initial set and \( \Gamma \) is a set of additional assumptions. Note that this rule indicates that ‘\( (L_1 \wedge \ldots \wedge L_n) \Rightarrow q \)’ is a formula in \( P_{N_i} \), and so ‘\( q \vdash L_1, \ldots, L_n \)’ is a clause in \( P_C \). If we convert the sequents in this rule, we
obtain the rule
\[
\frac{\Gamma \rightarrow L_1 \quad \cdots \quad \Gamma \rightarrow L_n}{\Gamma \rightarrow q},
\]
which we see is a valid SPRF Horn rule (given that \( q := L_1, \ldots, L_n \) is in \( P_c \)). The upper sequents of this instance of rule 3 are the roots of deductions with depth less than \( i \), so the induction hypothesis guarantees that the converted deductions (whose roots are the upper sequents of the Horn rule) are valid SPRF deductions. Hence, the entire converted refutation \( D^{T_2} \) is a valid SPRF deduction, i.e., \( P(i) \) holds.

**Lemma 2.2.** Let \( D \) be a SPRF deduction. Then \( D^{T_2^{-1}} \) is a valid condensed N-PROLOG deduction (as described above).

**Proof.** The proof follows by induction as in Lemma 2.1, with the roles of the systems reversed.

**Theorem 2.** There exists a bijective mapping between condensed N-PROLOG refutations (as described above) and SPRF refutations.

**Proof.** Let \( T_2 \) be the proof transformation described above, which removes the
initial set \( P_{N_i} \) from each sequent and replaces the symbol ‘?’ with ‘→’.

\[ \Rightarrow : \text{Let } R \text{ be a condensed N-PROLOG refutation. Since } R \text{ is also a condensed N-PROLOG deduction, it follows from Lemma 2.1 that } R^{T_2} \text{ is a valid SPRF deduction. And since the root of } R \text{ must be the sequent } 'P_{N_i} \rightarrow FALSE' \text{ (by definition of a condensed N-PROLOG refutation) which converts to the sequent } '→ FALSE', \text{ it follows that } R^{T_2} \text{ is also a valid SPRF refutation.} \]

\[ \leftarrow : \text{Similarly, let } R \text{ be a SPRF refutation. Since } R \text{ is also a SPRF deduction, it follows from Lemma 2.2 that } R^{T_2^{-1}} \text{ is a valid condensed N-PROLOG deduction. And since the root of } R \text{ must be the sequent } '→ FALSE' \text{ (by definition of a SPRF refutation) which converts to the sequent } 'P_{N_i} \rightarrow FALSE', \text{ it follows that } R^{T_2^{-1}} \text{ is also a valid condensed N-PROLOG refutation.} \]

**APPENDIX C**

We now prove Theorem 3 and Theorem 4, which formalize the relationship between N-PROLOG and SPRF-D/InH-PROLOG. As was noted in Appendix B, the rules of N-PROLOG, while presented in a linearized format, lend themselves to a proof-tree notation as well. We can rewrite the condensed N-PROLOG rules of Section 4.2 as follows:

**Rule 1.** Axiom: ‘\( P ? q \)’ where \( q \in P \).

**Rule 2.** For each formula of the form ‘\( [L_1 \land \cdots \land L_n \land (H_1 \rightarrow FALSE) \land \cdots \land (H_m \rightarrow FALSE)] \rightarrow q \)’ in \( P \) there is the inference rule
\[
\frac{P \land L_1 \quad \cdots \quad P \land L_n \quad P + H_1 \land FALSE \quad \cdots \quad P + H_m \land FALSE}{P \land q}.
\]

Let \( P_c \) denote the input set written in clausal form (i.e. as in SPRF-D and
InH-PROLOG), and let $P_{N_z}$ denote the input set written as described in Section 4.2 (where the clauses are represented using only conjunction and disjunction, and non-Horn clauses are represented by multiple formulas). A refutation in this system is a derivation of the sequent $'P_{N_z} \rightarrow FALSE'$ from axioms (instances of rule 1) via the generated inference rules (instances of rule 2).

As in the previous Appendices, we define a deduction (in any system) to be a derivation of a sequent from axioms via the inference rules of that system. We define the transformation $T_2$ on deductions as in Appendix B: $T_2$ removes the initial set $P_{N_z}$ from the antecedent of each sequent and replaces the symbol '?' with ' + '. We say that $D_{T_2}$ is the deduction obtained by applying $T_2$ to the deduction $D$. Likewise, $D_{T_2}^{-1}$ is the deduction obtained by applying the inverse transformation $T_2^{-1}$ to the deduction $D$.

**Lemma 3.1.** Let $D$ be a condensed N-PROLOG deduction (as described above). Then $D_{T_2}$ is a valid SPRF-D deduction.

**Proof** (By induction). We define the depth of a sequent in a deduction as follows: the depth of the root sequent is 1; if the lower sequent of a rule has depth $i$, then the upper sequents have depth $i + 1$. The depth of a deduction is the greatest depth of any sequent in the deduction. We now prove the lemma by induction using the following induction predicate:

$P(i)$. If the condensed N-PROLOG deduction $D$ has depth $i$, then $D_{T_2}$ is a valid SPRF-D deduction.

**Base:** $P(1)$. $D$ must be a tree with single sequent $'P_{N_z} + \Gamma \rightarrow ? 2'$ where $P_{N_z}$ is the initial set, $\Gamma$ is a set of additional assumptions, and $q$ is a fact in either $P_{N_z}$ (in which case the sequent is an instance of a degenerate rule 2, i.e. an axiom) or $\Gamma$ (in which case the sequent is an instance of the axiom, rule 1). When we convert this sequent, we obtain the sequent $'\Gamma \rightarrow q'$. If $q$ is a fact in the initial set $P_{N_z}$, then it is also a fact in $P$, so there is a degenerate SPRF-D Horn rule, i.e. axiom, of the form $'\Gamma \rightarrow q'$. If $q$ is an assumption in $\Gamma$, then $'\Gamma \rightarrow q'$ is an instance of the SPRF-D axiom. In either case, $D_{T_2}^{-1}$, which consists of this single sequent, is a valid SPRF-D deduction.

**Induction hypothesis:** Assume $P(k)$ for all $k < i$.

**Step:** Assume $D$ has depth $i$. The rule found at the root of $D$ must be an instance of rule 2 of the form

$$
\frac{P_{N_z} + L_1 \cdot \cdot \cdot P_{N_z} + L_n \cdot \cdot \cdot P_{N_z} + H_1 \rightarrow FALSE \cdot \cdot \cdot P_{N_z} + H_m \rightarrow FALSE}{P_{N_z} \rightarrow FALSE}
$$

where $P_{N_z}'$ is the initial set $P_{N_z}$ with additional assumptions $\Gamma$. Note that this rule indicates that $'(L_1 \land \cdot \cdot \cdot \land L_n \land (H_1 \rightarrow FALSE) \land \cdot \cdot \cdot \land (H_m \rightarrow FALSE)) \rightarrow q'$ is a formula in $P_{N_z}$, and so $'q; H_1; \ldots; H_m := L_1, \ldots, L_n'$ is a clause in $P$. If we convert the sequents in this rule, we obtain the rule

$$
\frac{\Gamma \rightarrow L_1 \cdot \cdot \cdot \Gamma \rightarrow L_n \cdot \cdot \cdot \Gamma, H_1 \rightarrow FALSE \cdot \cdot \cdot \Gamma, H_m \rightarrow FALSE}{\Gamma \rightarrow q}
$$
which we see is a valid SPRF-D rule (given that \( q; H_1; \ldots; H_m; \neg L_1, \ldots, L_n \) is in \( P_i \)). The upper sequents of this instance of rule 2 are the roots of deductions with depth less than \( i \), so the induction hypothesis guarantees that the converted deductions (whose roots are the upper sequents of the SPRF-D rule) are valid SPRF-D deductions. Hence, the entire converted refutation \( D^{T_2} \) is a valid SPRF-D deduction, i.e., \( P(i) \) holds.

**Lemma 3.2.** Let \( D \) be a SPRF-D deduction. Then \( D^{T_2^{-1}} \) is a valid condensed N-PROLOG deduction (as described above).

**Proof.** The proof follows by induction as in Lemma 3.1, with the roles of the systems reversed.

**Theorem 3.** There exists a bijective mapping between condensed N-PROLOG refutations (as described above) and SPRF-D refutations.

**Proof.** Let \( T_2 \) be the transformation described above, which removes the initial set \( P_{N_2} \) from each sequent and replaces the symbol '?' with '→'.

\[ \Rightarrow: \text{Let } R \text{ be a condensed N-PROLOG refutation. Since } R \text{ is also a condensed N-PROLOG deduction, it follows from Lemma 3.1 that } R^{T_2} \text{ is a valid SPRF-D deduction. And since the root of } R \text{ must be the sequent } P_{N_2} ? \text{FALSE}' \text{ (by definition of a condensed N-PROLOG refutation) which converts to the sequent } \rightarrow \text{FALSE}', \text{ it follows that } R^{T_2} \text{ is also a valid SPRF-D refutation.} \]

\[ \Leftarrow: \text{Similarly, let } R \text{ be a SPRF-D refutation. since } R \text{ is also a SPRF-D deduction, it follows from Lemma 3.2 that } R^{T_2^{-1}} \text{ is a valid condensed N-PROLOG deduction. And since the root of } R \text{ must be the sequent } P_{N_2} ? \text{FALSE}' \text{ (by definition of a SPRF-D refutation) which converts to the sequent } \rightarrow \text{FALSE}', \text{ it follows that } R^{T_2^{-1}} \text{ is also a valid condensed N-PROLOG refutation.} \]

**Theorem 4.** There exists a bijective mapping between condensed N-PROLOG refutations in which the cancellation pruning rule is enforced (i.e., every assumed literal is used in a rule (1) application) and InH-PROLOG refutations.

**Proof.** Let \( T_2 \) be the transformation described above, which removes the initial set from the antecedent of each sequent and replaces the symbol '?' with '→'. Theorem 3 proves that \( T_2 \) is a bijective mapping between condensed N-PROLOG refutations and SPRF-D refutations. Furthermore, it can be seen that if the cancellation pruning rule is enforced in a condensed N-PROLOG refutation \( R \), then it is also used in the corresponding SPRF-D refutation \( R^{T_2} \). Likewise, if the cancellation pruning rule is enforced in SPRF-D refutation \( R \), then it is used in condensed N-PROLOG refutation \( R^{T_2^{-1}} \).

Let \( T_1 \) be the transformation described in Appendix A, which removes InH-PROLOG sequents of the form \( (H)_{A_j, \ldots, A_j} \) and replaces each sequent of the form \( G \# A_j, \ldots, A_j \) with the sequent \( A_1, \ldots, A_j \rightarrow G \). Theorem 1 proves that \( T_1 \) is a bijective mapping between InH-PROLOG refutations and SPRF-D refutations in which the cancellation pruning rule is enforced (i.e., in which every subtree with root sequent of the form \( A_1, \ldots, A_j \rightarrow \text{FALSE} \) contains the axiom \( A_1, \ldots, A_j \rightarrow A_j \) as a leaf).
Thus, the composition $T_1^{-1} \circ T_2$ is a bijective mapping between condensed N-PROLOG refutations in which the cancellation pruning rule is enforced and InH-PROLOG refutations.

REFERENCES