Theorem Proving and Partial Proof Search for Intuitionistic Propositional Logic Using a Permutation-free Calculus with Loop Checking

Jacob M. Howe

December 5, 1996

Computer Science Division
University of St. Andrews, Scotland, KY16 9SS
jacob@dcs.st-and.ac.uk

1 Introduction

Backwards proof search and theorem proving with the standard cut-free sequent calculus for the propositional fragment of intuitionistic logic, Gentzen's LJ, is inefficient for three reasons. Firstly the proof search is not in general terminating, due to the possibility of looping. Secondly it will produce proofs which are essentially the same; they are permutations of each other, and correspond to the same natural deduction. Thirdly there are choice points where it has to be decided which of several rules to apply.

The sequent calculus MJ for intuitionistic logic was introduced (with another name, LJT) by Herbelin in [5]. This uses Girard's idea of a special place for formulae in the antecedent, the stoup first seen in [4]. The calculus was developed by Dyckhoff and Pinto ([1], [2]) because it has the property that proofs are in 1-1 correspondence with the normal natural deductions. MJ is a permutation-free sequent calculus; it avoids the problems of permutations that the cut-free sequent calculus of Gentzen has. The propositional fragment of the calculus MJ is displayed in Figure 1. This removes the second of the problems (and partly addresses the third). However, the naïve implementation of this calculus will lead to the possibility of looping.

$$\frac{A, \Gamma \Rightarrow B}{\Gamma \Rightarrow A \supset B} (\supset_{\mathcal{R}}) \qquad \frac{\Gamma \Rightarrow A \quad \Gamma \xrightarrow{B} C}{\Gamma \xrightarrow{A \supset B} C} (\supset_{\mathcal{L}})$$

$$\frac{\Gamma \xrightarrow{A} C}{\Gamma \xrightarrow{A \wedge B} C} (\wedge_{\mathcal{L}1}) \qquad \frac{\Gamma \xrightarrow{B} C}{\Gamma \xrightarrow{A \wedge B} C} (\wedge_{\mathcal{L}2})$$

$$\frac{\Gamma \Rightarrow A \quad \Gamma \Rightarrow B}{\Gamma \Rightarrow A \wedge B} (\wedge_{\mathcal{R}})$$

$$\frac{A, \Gamma \Rightarrow C \quad B, \Gamma \Rightarrow C}{\Gamma \xrightarrow{A \vee B} C} (\vee_{\mathcal{L}})$$

$$\frac{\Gamma \Rightarrow A}{\Gamma \Rightarrow A \vee B} (\vee_{\mathcal{R}1}) \qquad \frac{\Gamma \Rightarrow B}{\Gamma \Rightarrow A \vee B} (\vee_{\mathcal{R}2})$$

$$\frac{A, \Gamma \xrightarrow{A} B}{A, \Gamma \Rightarrow B} (C)$$

$$\frac{}{\Gamma \xrightarrow{A} A} (ax) \qquad \frac{}{\Gamma \xrightarrow{\perp} A} (\perp)$$

Define $\neg A \equiv A \supset \perp$.
A, B, C are formulae. Γ is a multiset of formulae.
A, Γ = $\{A\} \cup \Gamma$, where $\cup$ is multiset union.

Figure 1: Propositional Fragment of the calculus MJ.

Looping may easily be removed by checking whether a sequent has already occurred in a branch. Implementation of this is extremely inefficient as it requires too much information to be stored. Recent work by Heuerding et al [6] (the intuitionistic case of which is closely related to that of Gabbay in [3]) shows how to use a history to prevent looping in a far more efficient way.

In this paper the history mechanism is developed in two ways and applied to MJ. Both have advantages and disadvantages. The resulting calculi are then implemented in prolog to give efficient theorem proving and partial proof search. We call the new calculus MJ^{Hist}, the two varieties 'Swiss' and 'Scottish'.

2 The Use of Histories to Prevent Looping

Looping can very easily occur in MJ, for example:

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\vdots\\(p\wedge p)\supset p\Rightarrow p\end{array}\quad\begin{array}{c}\vdots\\(p\wedge p)\supset p\Rightarrow p\end{array}}{(p\wedge p)\supset p\Rightarrow p\wedge p}(\wedge_{\mathcal{R}})\quad\dfrac{}{(p\wedge p)\supset p\xrightarrow{p}p}(ax)}{(p\wedge p)\supset p\xrightarrow{(p\wedge p)\supset p}p}(\supset_{\mathcal{L}})}{(p\wedge p)\supset p\Rightarrow p}(C)$$

The sequent $(p\vee p)\supset p\Rightarrow p$ may continue to occur in the proof tree for this sequent using the MJ calculus. We can see that there is a loop, but we want to be able to automate this intuition. We need a way to detect and prevent such loops.

One way to do this is to add a *history* to each sequent. The history is the set of all sequents that have occurred so far in a proof tree. After each backwards inference the new sequent (without its history) is checked to see whether it is a member of this set. If it is we have looping and we backtrack. If not the new history is the union of the new sequent (without its history) and the old history, and we try to prove the new sequent, and so on. Unfortunately this method is very inefficient as it requires long lists of sequents to be stored by the computer, and all of this list has to be checked at each stage. When the sequents are stored we are keeping far more information than is necessary. It would be nice to cut down the amount of storage and checking to the bare minimum needed to prevent looping from occurring.

The basis of the reduced history is the realisation that one need only store goal formulae in order to loop-check. The rules of MJ are such that the context cannot decrease; once a formula is in the context it will remain in the context of all sequents above it in the proof tree. For two sequents to be the same they obviously need to have the same context. We may empty the history every time the context is extended, since we will never get any of the sequents below the extended one again. All we need store in the history are goal formulae. If we come across a goal already in the history we have the same goal and the same context as another sequent, that is, a loop.

There are two slightly different approaches to doing this. There is the straightforward extension and modification of the calculus described in [6] (which I shall call a Swiss history), and there is an approach which involves storing more formulae in the history, but that detects loops more quickly. This we will describe as the Scottish history, and the implementation is in some cases more efficient than the Swiss method. The calculus for the Swiss approach is displayed in Figure 2 and for the Scottish approach in Figure 3 (ε denotes the empty history).

$$\frac{A,\Gamma \Rightarrow B;\varepsilon}{\Gamma \Rightarrow A \supset B;\mathcal{H}}\ (\supset_{\mathcal{R}1}) \quad if A \notin \Gamma \qquad \frac{\Gamma \Rightarrow B;\mathcal{H}}{\Gamma \Rightarrow A \supset B;\mathcal{H}}\ (\supset_{\mathcal{R}2}) \quad if A \in \Gamma$$

$$\frac{A,\Gamma \Rightarrow \bot;\varepsilon}{\Gamma \Rightarrow \neg A;\mathcal{H}}\ (\neg_{\mathcal{R}1}) \quad if A \notin \Gamma \qquad \frac{\Gamma \Rightarrow \bot;\mathcal{H}}{\Gamma \Rightarrow \neg A;\mathcal{H}}\ (\neg_{\mathcal{R}2}) \quad if A \in \Gamma$$

$$\frac{\Gamma \Rightarrow A;(C,\mathcal{H}) \quad \Gamma \xrightarrow{B} C;\mathcal{H}}{\Gamma \xrightarrow{A\supset B} C;\mathcal{H}}\ (\supset_{\mathcal{L}}) \quad if C \notin \mathcal{H}$$

$$\frac{\Gamma \Rightarrow A;(C,\mathcal{H})}{\Gamma \xrightarrow{\neg A} C;\mathcal{H}}\ (\neg_{\mathcal{L}}) \quad if C \notin \mathcal{H}$$

$$\frac{\Gamma \xrightarrow{A} C;\mathcal{H}}{\Gamma \xrightarrow{A\wedge B} C;\mathcal{H}}\ (\wedge_{\mathcal{L}1}) \qquad \frac{\Gamma \xrightarrow{B} C;\mathcal{H}}{\Gamma \xrightarrow{A\wedge B} C;\mathcal{H}}\ (\wedge_{\mathcal{L}2})$$

$$\frac{\Gamma \Rightarrow A;\mathcal{H} \quad \Gamma \Rightarrow B;\mathcal{H}}{\Gamma \Rightarrow A \wedge B;\mathcal{H}}\ (\wedge_{\mathcal{R}})$$

$$\frac{A,\Gamma \Rightarrow C;\varepsilon \quad B,\Gamma \Rightarrow C;\varepsilon}{\Gamma \xrightarrow{A\vee B} C;\mathcal{H}}\ (\vee_{\mathcal{L}}) \quad if A,B \notin \Gamma$$

$$\frac{\Gamma \Rightarrow A;\mathcal{H}}{\Gamma \Rightarrow A \vee B;\mathcal{H}}\ (\vee_{\mathcal{R}1}) \qquad \frac{\Gamma \Rightarrow B;\mathcal{H}}{\Gamma \Rightarrow A \vee B;\mathcal{H}}\ (\vee_{\mathcal{R}2})$$

$$\frac{A,\Gamma \xrightarrow{A} B;\mathcal{H}}{A,\Gamma \Rightarrow B;\mathcal{H}}\ (C)^{\star} \qquad \frac{}{\Gamma \xrightarrow{A} A;\mathcal{H}}\ (ax) \qquad \frac{}{\Gamma \xrightarrow{\bot} A;\mathcal{H}}\ (\bot)$$

$\star$ B is either a propositional variable, $\bot$ or a disjunction.
A, B, C are formulae. Γ, $\mathcal{H}$ are sets of formulae.
B, Γ is shorthand for $\{B\} \cup \Gamma$.

Figure 2: The propositional calculus MJ^{Hist}, Swiss style

$$\frac{A, \Gamma \Rightarrow B; \{B\}}{\Gamma \Rightarrow A \supset B; \mathcal{H}} (\supset_{\mathcal{R}1}) \quad if A \notin \Gamma \qquad \frac{A, \Gamma \Rightarrow \bot; \{\bot\}}{\Gamma \Rightarrow \neg A; \mathcal{H}} (\neg_{\mathcal{R}1}) \quad if A \notin \Gamma$$

$$\frac{\Gamma \Rightarrow B; (B, \mathcal{H})}{\Gamma \Rightarrow A \supset B; \mathcal{H}} (\supset_{\mathcal{R}2}) \quad if A \in \Gamma, \quad if B \notin \mathcal{H}$$

$$\frac{\Gamma \Rightarrow \bot; (\bot, \mathcal{H})}{\Gamma \Rightarrow \neg A; \mathcal{H}} (\neg_{\mathcal{R}2}) \quad if A \in \Gamma, \quad if \bot \notin \mathcal{H}$$

$$\frac{\Gamma \Rightarrow A; (A, \mathcal{H}) \quad \Gamma \xrightarrow{B} C; \mathcal{H}}{\Gamma \xrightarrow{A \supset B} C; \mathcal{H}} (\supset_{\mathcal{L}}) \quad if A \notin \mathcal{H}$$

$$\frac{\Gamma \Rightarrow A; (A, \mathcal{H})}{\Gamma \xrightarrow{\neg A} C; \mathcal{H}} (\neg_{\mathcal{L}}) \quad if A \notin \mathcal{H}$$

$$\frac{\Gamma \xrightarrow{A} C; \mathcal{H}}{\Gamma \xrightarrow{A \wedge B} C; \mathcal{H}} (\wedge_{\mathcal{L}1}) \qquad \frac{\Gamma \xrightarrow{B} C; \mathcal{H}}{\Gamma \xrightarrow{A \wedge B} C; \mathcal{H}} (\wedge_{\mathcal{L}2})$$

$$\frac{\Gamma \Rightarrow A; (A, \mathcal{H}) \quad \Gamma \Rightarrow B; B, \mathcal{H}}{\Gamma \Rightarrow A \wedge B; \mathcal{H}} (\wedge_{\mathcal{R}}) \quad if A, B \notin \mathcal{H}$$

$$\frac{A, \Gamma \Rightarrow C; \{C\} \quad B, \Gamma \Rightarrow C; \{C\}}{\Gamma \xrightarrow{A \vee B} C; \mathcal{H}} (\vee_{\mathcal{L}}) \quad if A, B \notin \Gamma$$

$$\frac{\Gamma \Rightarrow A; (A, \mathcal{H})}{\Gamma \Rightarrow A \vee B; \mathcal{H}} (\vee_{\mathcal{R}1}) \quad if A \notin \mathcal{H} \qquad \frac{\Gamma \Rightarrow B; (B, \mathcal{H})}{\Gamma \Rightarrow A \vee B; \mathcal{H}} (\vee_{\mathcal{R}2}) \quad if B \notin \mathcal{H}$$

$$\frac{A, \Gamma \xrightarrow{A} B; \mathcal{H}}{A, \Gamma \Rightarrow B; \mathcal{H}} (C)^{\star} \qquad \frac{}{\Gamma \xrightarrow{A} A; \mathcal{H}} (ax) \qquad \frac{}{\Gamma \xrightarrow{\bot} A; \mathcal{H}} (\bot)$$

$\star$ B is either a propositional variable, $\bot$ or a disjunction.
A, B, C are formulae. $\Gamma, \mathcal{H}$ are sets of formulae.
B, Γ is shorthand for $\{B\} \cup \Gamma$.

Figure 3: The propositional calculus MJ^{Hist}, Scottish

3 Theorem Proving Using the Swiss History

In this section we shall first look at the theory of the Swiss calculus, and then we shall look at an implementation of this calculus in prolog. Before continuing, we should point out that the calculi we describe as Swiss are significantly different from the one in [6]. This is partly because of the stoup in MJ and partly because as we are trying to focus on the history mechanism, we have not included the subsumption checks that the calculus in [6] uses.

3.1 Theory for MJ^{Hist}, Swiss style

First note that whereas in MJ negation was defined via the equivalence $\neg A \equiv A \supset \bot$, in MJ^{Hist} it has its own rules. These rules are just special cases of the rules for $\supset$. For the case of $(\neg_{\mathcal{L}})$ there is only one premiss since the other that would be produced by $(\supset_{\mathcal{L}})$ is an instance of $(\bot)$, and is not needed.

Also notice that there are two rules for $(\supset_{\mathcal{R}})$. These correspond to the two cases where the formula is and isn't in the context. As noted above this is very important for MJ^{Hist}. Finally note that the context is a set, not a multiset. We may only have one copy of a formula in the context, but the formula in the stoup may also be in the context.

The calculus works in a similar way to that of [6]. Right rules are applied to a sequent until the goal formula is either a propositional variable, falsum, or a disjunction (note that disjunction isn't covered in [6], and requires special treatment). Then a formula from the context is selected and placed in the stoup by the (C) rule, and left rules are applied to it (this focussing obviously does not occur in [6]). The history loop checking occurs in the $(\supset_{\mathcal{L}})$ rule. The left premiss of this rule has the same context as the conclusion, but the goal is likely to be different. We store the goal of the conclusion in the history and then continue backwards proof search on the left premiss. The loop detection also occurs at this same point. If the goal of the conclusion of the $(\supset_{\mathcal{L}})$ rule is already in the history, then there is a loop (the context can't have been extended and the goal is the same as in some other sequent), and so this branch is not pursued. We look elsewhere for a proof. Likewise for $(\neg_{\mathcal{L}})$.

There is another place where the rules are restricted in order to prevent looping. This is the condition placed on the $(\vee_{\mathcal{L}})$ rule. For the $(\supset_{\mathcal{R}})$ rule (which attempts to extend the context) there are two cases corresponding to when the context is and isn't extended. Something similar is happening for the $(\vee_{\mathcal{L}})$ rule. In both the premisses of the rule a formula may be added to the context. If both contexts really are extended, then we continue building the proof tree. If one or both contexts are not extended then the sequent with the non-extended context will be the same as some sequent at a lesser height in the proof tree. That is, there is a loop. This is easy to see: the context and goal of this premiss are the same as that of the conclusion, the sequent before the stoup formula (or a formula containing it as a subformula) was selected into the stoup must be the same as the premiss sequent.

Next we need to prove that MJ^{Hist} and MJ are equivalent, that is, they prove the same sequents. First we prove that MJ^{Hist} without restriction $\star$ on the (C) rule is equivalent to MJ, then we shall prove the admissibility of the restriction.

First we need definitions for the measures that we will use in inductive proofs.

Definition 1 *The* **height** *of a sequent in a proof tree is defined as follows: The root of the tree has height 0. If sequent T is a premiss of rule (R) with conclusion S of height n, then sequent T has height n+1.*

Definition 2 *The* **depth** *of a proof is defined as follows: The leaves of the proof tree have depth 1. If sequent T is obtained from sequent S with proof of depth n by forward application of a one premiss rule, then the proof of T has depth n+1. If sequent T is obtained from sequent S with proof of depth n and sequent R with proof of depth m by forward application of a two premiss rule, then the proof of T has depth max(n, m)+1.*

Definition 3 *The* **complexity** *of a formula is defined as follows: Let c(F) denote the complexity of formula F. If X is a propositional variable, then c(X)=1. Also, c($\perp$)=1. Then define inductively, c(A $\wedge$ B)=c(A)+c(B), c(A $\vee$ B)=c(A)+c(B), c(A$\supset$B)=c(A)+c(B)+1, c($\neg$A)=c(A)+2.*

Lemma 1 *Weakening is an admissible rule of MJ. That is, if $\Gamma \Rightarrow B$ (or $\Gamma \xrightarrow{S} B$) is provable in MJ then $A, \Gamma \Rightarrow B$ (A, $\Gamma \xrightarrow{S} B$) is too. i.e.*

$$\frac{\Gamma \Rightarrow B}{A, \Gamma \Rightarrow B}\,(W) \qquad \frac{\Gamma \xrightarrow{S} B}{A, \Gamma \xrightarrow{S} B}\,(W)$$

are admissible.

PROOF: An easy induction. Omitted. ■

Theorem 1 *The systems MJ and MJ^{Hist} (without $\star$) are equivalent. That is, a sequent S is provable in MJ if and only if $S; \varepsilon$ (the sequent with empty history) is provable in MJ^{Hist} (without $\star$).*

PROOF: *a) If $S; \varepsilon$ is provable in MJ^{Hist} then S is provable in MJ.* This result is trivial. Given the proof in MJ^{Hist}, simply strip the history from each sequent of the proof tree. Where before we had an MJ^{Hist} inference we now have a valid MJ inference. The only exception to this is where the MJ^{Hist} rule is $(\supset_{\mathcal{R}2})$ (and analogously $(\neg_{\mathcal{R}})$). However, by noting that an implicit weakening has occurred at this point we can (by adding an extra step) make a valid MJ inference of the form required. We transform the MJ^{Hist} inference to an MJ one as follows:

$$\frac{\Gamma', A \Rightarrow B; \mathcal{H}}{\Gamma', A \Rightarrow A \supset B; \mathcal{H}}\ (\supset_{\mathcal{R}2})$$

is transformed to

$$\cfrac{\cfrac{\Gamma', A \Rightarrow B}{\Gamma', A, A \Rightarrow B}\ (W)}{\Gamma', A \Rightarrow A \supset B}\ (\supset_{\mathcal{R}})$$

completing the result.

b) If S is provable in MJ then $S; \varepsilon$ is provable in MJ^{Hist}. Take any proof of S in MJ. By definition of a proof in MJ the proof tree is finite, that is, all branches of the tree terminate with success at a finite height. Using a successful proof tree for sequent S in MJ we construct a successful proof tree for sequent $S; \varepsilon$ in MJ^{Hist}.

We need a notion of correspondence between branches of fragments of MJ and MJ^{Hist} proof trees: The root of an MJ tree corresponds to the root of an MJ^{Hist} proof tree if (when multisets are considered as sets, and the history is ignored) they are the same sequent. If branch A^{MJ} and branch A^{MJH} correspond, then if one branch is extended by backward application of a one premiss rule, and the other is extended by the same rule in its system then the two extended branches correspond. Likewise for a two premiss rule, where we have two new branches with the obvious correspondences. If a proof tree is altered by cutting bits out, then the new MJ^{Hist} branch will correspond to the MJ branch following the appropriate premiss. An MJ and an MJ^{Hist} proof tree correspond if every unfinished branch of the MJ^{Hist} tree corresponds to an unfinished branch of the MJ tree, and all unfinished branches of the MJ tree correspond to unfinished branches of the MJ^{Hist} tree.

We start the MJ^{Hist} proof tree with root $S; \varepsilon$, corresponding to the root S of the MJ proof tree. Given a fragment of an MJ^{Hist} proof tree corresponding to a fragment of the given MJ proof tree, we look at a branch, B, which is unfinished in the fragment of the MJ proof tree we are looking at. This will have a top sequent S at height *n*. We look at branch B' of the MJ tree, which is B extended by backward application of the next rule (R). B' has top sequent (or sequents) that are at height *n+1*. We now build a fragment of an MJ^{Hist} proof tree corresponding to a larger fragment of the MJ proof tree.

If the next rule on the MJ branch (i.e. (R)) is such that the corresponding rule in MJ^{Hist} is one without side conditions (i.e. $(\wedge_{\mathcal{R}})$, $(\wedge_{\mathcal{L}1})$, $(\wedge_{\mathcal{L}2})$, $(\vee_{\mathcal{R}1})$, $(\vee_{\mathcal{R}2})$, (ax), $(\perp)$, (C)) then the MJ^{Hist} proof tree corresponding to the extended proof tree in MJ (i.e. with B' instead of B) is gained simply by applying the appropriate rule to the appropriate branch. For example, if the next inference in the MJ proof tree is:

$$\frac{\Gamma \Rightarrow A \quad \Gamma \Rightarrow B}{\Gamma \Rightarrow A \wedge B}\ (\wedge_{\mathcal{R}})$$

then the next inference in the MJ^{Hist} is:

$$\frac{\Gamma \Rightarrow A; \mathcal{H} \quad \Gamma \Rightarrow B; \mathcal{H}}{\Gamma \Rightarrow A \wedge B; \mathcal{H}}\ (\wedge_{\mathcal{R}})$$

The cases with side conditions are $(\supset_{\mathcal{R}})$, $(\neg_{\mathcal{R}})$, $(\vee_{\mathcal{L}})$, $(\supset_{\mathcal{L}})$ and $(\neg_{\mathcal{L}})$. The negation rules can be treated analogously to the implication rules, and are discussed no further. For the $(\supset_{\mathcal{R}})$ rule we simply apply the version appropriate, depending on the context. (The possibility that the MJ sequents will have different contexts due to multiple occurrences of formulae is irrelevant to this process). For the $(\supset_{\mathcal{L}})$ and $(\vee_{\mathcal{L}})$ rules, if the side conditions are satisfied, then we simply build the MJ^{Hist} proof tree by applying the appropriate rule. If the side conditions are not satisfied it means that there is a loop and more work needs to be done.

We consider first the $(\vee_{\mathcal{L}})$ rule. When the conditions aren't met we have to remove bits from the MJ^{Hist} proof tree rather than add them. Note that in a sequent $\Gamma \xrightarrow{A\vee B} C; \mathcal{H}$ the formula $A \vee B$ must be a subformula of some formula F which was placed in the stoup by rule (C). This is easily seen by inspection of the rules of MJ^{Hist}. (On backwards application of rules to an unstouped sequent, a formula may only enter the stoup via (C). Suppose there is a sequent at height $n+1$ with stoup S. Then for any sequent at height n (with stoup T) it may follow from, S is a subformula of T).

Suppose that $A, B \in \Gamma$. We cannot apply the $(\vee_{\mathcal{L}})$ rule to the top MJ^{Hist} sequent. But if we did we would have:

$$\frac{\Gamma \Rightarrow C \quad \Gamma \Rightarrow C}{\Gamma \xrightarrow{A\vee B} C; \mathcal{H}}\ (\vee_{\mathcal{L}})$$

Consider the application of (C) which placed F in the stoup:

$$\frac{\Delta, F \xrightarrow{F} G; \mathcal{H}}{\Delta, F \Rightarrow G; \mathcal{H}}\ (C)$$

What are Δ and G? As we are supposing that all the sequents in the proof tree between $\Delta, F \xrightarrow{F} G; \mathcal{H}$ and $\Gamma \xrightarrow{A\vee B} C$ are stouped, then inspection of rules immediately gives us that $\Delta, F = \Gamma$ and that $G = C$. So in fact $\Gamma \Rightarrow C$ is $\Delta, F \Rightarrow G$. We are trying to prove the same sequent. So we cut off and throw away all of the MJ^{Hist} proof tree above $\Delta, F \Rightarrow G; \mathcal{H}$. This fragment of MJ^{Hist} proof tree corresponds to a fragment of the MJ tree. It is obviously a valid MJ^{Hist} tree fragment, and we continue to build on it, following, in this case, either branch of the MJ proof. In throwing away part of the proof tree we may have thrown away some unfinished branches. The new MJ^{Hist} proof tree fragment corresponds to the old MJ fragment with B extended to B' and with all branches corresponding to cut out MJ^{Hist} branches (and any branches above) finished (i.e. all subsequent branches end with leaves that are successful axiom or $(\bot)$ rules).

Now suppose that $A \in \Gamma$ and that $B \notin \Gamma$. Again the $(\vee_{\mathcal{L}})$ rule is not applicable. If we applied it we would have:

$$\frac{\Gamma \Rightarrow C \quad \Gamma, B \Rightarrow C}{\Gamma \xrightarrow{A\vee B} C; \mathcal{H}}\,(\vee_{\mathcal{L}})$$

We have an almost identical situation to the previous one and by chopping off the same part of the proof as before we get a valid MJ^{Hist} branch in the same way. We now follow the MJ branch above the left premiss. Obviously the $A \notin \Gamma$, $B \in \Gamma$ case is similar, but in this case we follow the right premiss in MJ.

Finally we consider the $(\supset_{\mathcal{L}})$ rule. This is the rule in which the history mechanism does something to prevent looping. If the history condition isn't met, it tells us that the part of the MJ^{Hist} proof tree built so far has the form (with $C \in \mathcal{H}'$, i.e. the history is not reset):

$$\begin{array}{c} \dfrac{\begin{array}{c}\dfrac{\Gamma \Rightarrow A \quad \Gamma \xrightarrow{B} C}{\Gamma \xrightarrow{A\supset B} C; \mathcal{H}'}\,(\supset_{\mathcal{L}}) \\ \vdots \\ \Gamma \Rightarrow C; \mathcal{H}' \\ \vdots \\ \Gamma \Rightarrow X; C, \mathcal{H}\end{array} \qquad \vdots}{\Gamma \xrightarrow{X\supset Y} C; \mathcal{H}}\,(\supset_{\mathcal{L}}) \quad C \notin \mathcal{H} \\ \vdots \\ \Gamma \Rightarrow C; \mathcal{H}\end{array}$$

This contains the loop which is the reason why the condition on the rule failed to be satisfied. The new branch in the MJ^{Hist} proof tree is formed by removing all the sequents above $\Gamma \Rightarrow C; \mathcal{H}$ up to and including the sequent $\Gamma \Rightarrow C; \mathcal{H}'$. The occurrences of $\mathcal{H}'$ in the tree above this point are replaced by occurrences of $\mathcal{H}$. It is seen that where before $(\supset_{\mathcal{L}})$ failed it will now succeed as the condition on the rule is met. The new branch is valid in MJ^{Hist} and the new tree corresponds to the extended MJ tree with the branches corresponding to those cut out being finished.

As has been noted, MJ proof trees are finite and at each stage we have a MJ^{Hist} proof tree corresponding to strictly more of the MJ tree. The process of building the MJ^{Hist} proof tree has only a finite number of stages and so must come to end. At the end of each branch (or at each leaf, if you like) of the MJ proof tree, there is an application of (ax) or $(\bot)$ (since the tree is, by assumption, a success). Therefore the last rule of each branch of the MJ^{Hist} proof tree must also be (ax) or $(\bot)$. Hence the proof tree is a success and we have a proof of the sequent in MJ^{Hist}. ■

Lemma 2 *The calculus MJ with condition $*$ placed on rule (C) is equivalent to MJ (without the extra condition).*

PROOF: a)*If sequent S is provable in MJ with $*$ then it is provable in MJ without* $*$. Trivial.

b)*If sequent S is provable in MJ without $*$ then it is provable in MJ with* $*$. The only inferences that we can make in the calculus without $*$ that we cannot make in the calculus with $*$ are contractions with implication, negation or conjunction as the goal. Negation is a special case of implication and will be discussed no further.

The proof is by induction on the depth of the proof and the complexity of formulae.

i) The formula selected for the stoup is a propositional variable. This case is not possible, for if it did happen, we could not have an axiom as the premiss and no rule is applicable to a sequent with a propositional variable in the stoup.

ii) The formula selected for the stoup is $\bot$.

$$\dfrac{\dfrac{}{\Gamma, \bot \xrightarrow{\bot} A \supset B}\,(\bot)}{\Gamma, \bot \Rightarrow A \supset B}\,(C) \qquad \dfrac{\dfrac{}{\Gamma, \bot \xrightarrow{\bot} A \wedge B}\,(\bot)}{\Gamma, \bot \Rightarrow A \wedge B}\,(C)$$

If the above are valid then the following are too:

$$\dfrac{\dfrac{\dfrac{}{A, \Gamma, \bot \xrightarrow{\bot} B}\,(\bot)}{A, \Gamma, \bot \Rightarrow B}\,(C)}{\Gamma, \bot \Rightarrow A \supset B}\,(\supset_{\mathcal{R}}) \qquad \dfrac{\dfrac{\dfrac{}{\Gamma, \bot \xrightarrow{\bot} A}\,(\bot)}{\Gamma, \bot \Rightarrow A}\,(C) \quad \dfrac{\dfrac{}{\Gamma, \bot \xrightarrow{\bot} B}\,(\bot)}{\Gamma, \bot \Rightarrow B}\,(C)}{\Gamma, \bot \Rightarrow A \wedge B}\,(\wedge_{\mathcal{R}})$$

By induction on the complexity of the goals of the conclusions of (C) we get the result.

iii) The formula selected for the stoup is a disjunction.

$$\dfrac{\dfrac{X, \Gamma, X \vee Y \Rightarrow A \supset B \quad Y, \Gamma, X \vee Y \Rightarrow A \supset B}{\Gamma, X \vee Y \xrightarrow{X \vee Y} A \supset B}\,(\vee_{\mathcal{L}})}{\Gamma, X \vee Y \Rightarrow A \supset B}\,(C)$$

and

$$\cfrac{\cfrac{X,\Gamma,X\vee Y\Rightarrow A\wedge B\quad Y,\Gamma,X\vee Y\Rightarrow A\wedge B}{\Gamma,X\vee Y\xrightarrow{X\vee Y}A\wedge B}\;(\vee_{\mathcal{L}})}{\Gamma,X\vee Y\Rightarrow A\wedge B}\;(C)$$

By induction on the depth of the proof we have MJ with $\star$ proofs of:

$$X,\Gamma,X\vee Y\Rightarrow A\supset B\qquad Y,\Gamma,X\vee Y\Rightarrow A\supset B$$

$$X,\Gamma,X\vee Y\Rightarrow A\wedge B\qquad Y,\Gamma,X\vee Y\Rightarrow A\wedge B$$

Therefore we have proofs of the above sequents whose next step are right rules. And so we have:

$$\cfrac{\cfrac{\cfrac{A,X,\Gamma,X\vee Y\Rightarrow B\quad A,Y,\Gamma,X\vee Y\Rightarrow B}{A,\Gamma,X\vee Y\xrightarrow{X\vee Y}B}\;(\vee_{\mathcal{L}})}{A,\Gamma,X\vee Y\Rightarrow B}\;(C)}{\Gamma,X\vee Y\Rightarrow A\supset B}\;(\supset_{\mathcal{R}})$$

and (where $\Gamma' = \Gamma, X\vee Y$)

$$\cfrac{\cfrac{\cfrac{X,\Gamma'\Rightarrow A\quad Y,\Gamma'\Rightarrow A}{\Gamma,X\vee Y\xrightarrow{X\vee Y}A}\;(\vee_{\mathcal{L}})}{\Gamma,X\vee Y\Rightarrow A}\;(C)\qquad\cfrac{\cfrac{X,\Gamma'\Rightarrow B\quad Y,\Gamma'\Rightarrow B}{\Gamma,X\vee Y\xrightarrow{X\vee Y}B}\;(\vee_{\mathcal{L}})}{\Gamma,X\vee Y\Rightarrow B}\;(C)}{\Gamma,X\vee Y\Rightarrow A\wedge B}\;(\wedge_{\mathcal{R}})$$

And by induction on the complexity of the goals of the conclusions of the (C) rule we get the result.

Alternatively we could have axioms:

$$\cfrac{\cfrac{\cfrac{}{A,\Gamma,X\vee Y\xrightarrow{X\vee Y}B}\;(ax)}{A,\Gamma,X\vee Y\Rightarrow B}\;(C)}{\Gamma,X\vee Y\Rightarrow A\supset B}\;(\supset_{\mathcal{R}})\qquad\cfrac{\cfrac{\begin{matrix}\vdots\\ \Gamma,X\vee Y\xrightarrow{X\vee Y}A\end{matrix}}{\Gamma,X\vee Y\Rightarrow A}\;(C)\qquad\cfrac{\cfrac{}{\Gamma,X\vee Y\xrightarrow{X\vee Y}B}\;(ax)}{\Gamma,X\vee Y\Rightarrow B}\;(C)}{\Gamma,X\vee Y\Rightarrow A\wedge B}\;(\wedge_{\mathcal{R}})$$

And we get the result by induction on the complexity of the goal formulae in the contractions.

iv) The formula selected for the stoup is a conjunction.

$$\cfrac{\cfrac{\Gamma,X\wedge Y\xrightarrow{X}A\supset B}{\Gamma,X\wedge Y\xrightarrow{X\wedge Y}A\supset B}\;(\wedge_{\mathcal{L}})}{\Gamma,X\wedge Y\Rightarrow A\supset B}\;(C)\qquad\cfrac{\cfrac{\Gamma,X\wedge Y\xrightarrow{X}A\wedge B}{\Gamma,X\wedge Y\xrightarrow{X\wedge Y}A\wedge B}\;(\wedge_{\mathcal{L}})}{\Gamma,X\wedge Y\Rightarrow A\wedge B}\;(C)$$

It is possible that the top sequent is an axiom, that is:

$$\dfrac{}{\Gamma, X \wedge Y \xrightarrow{A \supset B} A \supset B}\,(ax) \qquad \dfrac{}{\Gamma, X \wedge Y \xrightarrow{A \wedge B} A \wedge B}\,(ax)$$

in which case we have the following (where X = $A \supset B$):

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A, \Gamma, X \wedge Y \xrightarrow{A} A}\,(ax)}{A, \Gamma, X \wedge Y \Rightarrow A}\,(C) \quad \dfrac{}{A, \Gamma, X \wedge Y \xrightarrow{B} B}\,(ax)}{A, \Gamma, X \wedge Y \xrightarrow{A \supset B} B}\,(\supset_{\mathcal{L}})}{A, \Gamma, X \wedge Y \xrightarrow{X \wedge Y} B}\,(\wedge_{\mathcal{L}})}{A, \Gamma, X \wedge Y \Rightarrow B}\,(C)}{\Gamma, X \wedge Y \Rightarrow A \supset B}\,(\supset_{\mathcal{R}})$$

and (where X = $A \wedge B$)

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma, X \wedge Y \xrightarrow{A} A}\,(ax)}{\Gamma, X \wedge Y \xrightarrow{A \wedge B} A}\,(\wedge_{\mathcal{L}})}{\Gamma, X \wedge Y \xrightarrow{X \wedge Y} A}\,(\wedge_{\mathcal{L}})}{\Gamma, X \wedge Y \Rightarrow A}\,(C) \quad \dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma, X \wedge Y \xrightarrow{B} B}\,(ax)}{\Gamma, X \wedge Y \xrightarrow{A \wedge B} B}\,(\wedge_{\mathcal{L}})}{\Gamma, X \wedge Y \xrightarrow{X \wedge Y} B}\,(\wedge_{\mathcal{L}})}{\Gamma, X \wedge Y \Rightarrow B}\,(C)}{\Gamma, X \wedge Y \Rightarrow A \wedge B}\,(\wedge_{\mathcal{R}})$$

In these cases we get the result by induction on the complexity of the goal formula in the contractions.

If we don't have axioms, the result follows by induction on the complexity of the stoup formula.

v) The formula selected for the stoup is an implication.

$$\dfrac{\dfrac{\Gamma, X \supset Y \Rightarrow X \quad \Gamma, X \supset Y \xrightarrow{Y} A \supset B}{\Gamma, X \supset Y \xrightarrow{X \supset Y} A \supset B}\,(\supset_{\mathcal{L}})}{\Gamma, X \supset Y \Rightarrow A \supset B}\,(C)$$

or

$$\dfrac{\dfrac{\Gamma, X \supset Y \Rightarrow X \quad \Gamma, X \supset Y \xrightarrow{Y} A \wedge B}{\Gamma, X \supset Y \xrightarrow{X \supset Y} A \wedge B}\,(\supset_{\mathcal{L}})}{\Gamma, X \supset Y \Rightarrow A \wedge B}\,(C)$$

By the weakening lemma for MJ we may prove $A, \Gamma, X \supset Y \Rightarrow X$. It is possible that the right premiss is an axiom, that is:

$$\dfrac{}{\Gamma, X \supset Y \xrightarrow{A \supset B} A \supset B}\,(ax) \qquad \dfrac{}{\Gamma, X \supset Y \xrightarrow{A \wedge B} A \wedge B}\,(ax)$$

In which case we have (where Γ' = $\Gamma, X \supset Y$ and Y = $A \supset B$):

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\vdots\\ A, \Gamma' \Rightarrow X\end{array} \qquad \dfrac{\dfrac{\dfrac{}{A, \Gamma' \xrightarrow{A} A}\,(ax)}{A, \Gamma' \Rightarrow A}\,(C) \qquad \dfrac{}{A, \Gamma' \xrightarrow{B} B}\,(ax)}{A, \Gamma' \xrightarrow{A \supset B} B}\,(\supset_{\mathcal{L}})}{A, \Gamma' \xrightarrow{X \supset Y} B}\,(\supset_{\mathcal{L}})}{A, \Gamma' \Rightarrow B}\,(C)}{\Gamma' \Rightarrow A \supset B}\,(\supset_{\mathcal{R}})$$

and (again with Γ' = $\Gamma, X \supset Y$ and Y = $A \wedge B$):

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\vdots\\ \Gamma' \Rightarrow X\end{array} \quad \dfrac{\dfrac{}{\Gamma' \xrightarrow{A} A}\,(ax)}{\Gamma' \xrightarrow{A \wedge B} A}\,(\wedge_{\mathcal{L}})}{\Gamma' \xrightarrow{X \supset Y} A}\,(\supset_{\mathcal{L}})}{\Gamma' \Rightarrow A}\,(C) \qquad \dfrac{\dfrac{\begin{array}{c}\vdots\\ \Gamma' \Rightarrow X\end{array} \quad \dfrac{\dfrac{}{\Gamma' \xrightarrow{B} B}\,(ax)}{\Gamma' \xrightarrow{A \wedge B} B}\,(\wedge_{\mathcal{L}})}{\Gamma' \xrightarrow{X \supset Y} B}\,(\supset_{\mathcal{L}})}{\Gamma' \Rightarrow B}\,(C)}{\Gamma' \Rightarrow A \wedge B}\,(\wedge_{\mathcal{R}})$$

We get the result by induction on the complexity of the goal formulae in the contractions.

If we do not have the axiom case we get the result by induction on the complexity of the stoup formula. ■

Theorem 2 *The calculus MJ^{Hist} with condition $*$ placed on rule (C) is equivalent to MJ^{Hist} without the extra condition.*

PROOF: a)*If a sequent S is provable in MJ^{Hist} with $*$ then it is provable in MJ^{Hist} without $*$.* Trivial.

b)*If sequent S is provable in MJ^{Hist} without $*$ then it is provable in MJ^{Hist} with $*$.* Given a proof of $\Gamma \Rightarrow A; \mathcal{H}$ in MJ^{Hist} without $*$, strip away the history (as in the proof of Theorem 1a)). We then have an MJ proof of $\Gamma \Rightarrow A$. By Lemma 2 we have a proof of $\Gamma \Rightarrow A$ in MJ with $*$. By the construction in Theorem 1b) we get a proof of $\Gamma \Rightarrow A; \varepsilon$ in MJ^{Hist} with $*$. ■

3.2 Implementation of MJ^{Hist} in the Swiss Style

In this section we give an implementation of MJ^{Hist} with a Swiss history. All sets are represented by lists. The implementation is in SICStus Prolog. The code uses an abstract syntax so that translations from other systems can be achieved using only a small program on top of the given code.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% We need to use the prolog library use_module(library(lists)).
% for the predicates member and non_member.
%
% Operators are defined for each of the connectives
% of the calculus.  Propositional variables are
% given a prefix operator pv().
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

:-op(500, xfy, imp).
:-op(500, yfx, and).
:-op(500, yfx, or).
:-op(300, fx, not).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The two place predicate prove will attempt to
% prove non-stouped sequents, Context => Goal.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

prove(G, Context):-
         provenostoup(G, Context, []).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The three place predicate provenostoup deals with
% sequents with a history, but no stoup,
% Context => Goal ; History.  Using first argument indexing we
% first deal with implications, negations and conjunctions,
% with the appropriate checks.  Next the three cases of the
% contraction rule with the restriction implemented.  Finally
% we have the disjunction rules, whose application may
% necessarily have to be delayed, and hence are last.
% The predicate takes arguments
% provenostoup(Goal, Context, History)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

provenostoup(A imp B, Context, _):-
          non_member(A, Context), !,
          provenostoup(B, [A | Context], []), !.

provenostoup(A imp B, Context, History):-
          member(A, Context), !,
          provenostoup(B, Context, History), !.

provenostoup(not(A), Context, History):-
```

```
          non_member(A, Context), !,
          provenostoup(falsum, [A | Context], []), !.

provenostoup(not(A), Context, History):-
          member(A, Context), !,
          provenostoup(falsum, Context, History), !.

provenostoup(A and B, Context, History):-
          provenostoup(A, Context, History), !,
          provenostoup(B, Context, History), !.

provenostoup(pv(A), Context, History):-
          member(X, Context),
          provestoup(X, pv(A), Context, History), !.

provenostoup(falsum, Context, History):-
          member(X, Context),
          provestoup(X, falsum, Context, History), !.

provenostoup(A or B, Context, History):-
          member(X, Context),
          provestoup(X, A or B, Context, History), !.

provenostoup(A or B, Context, History):-
          provenostoup(A, Context, History), !.

provenostoup(A or B, Context, History):-
          provenostoup(B, Context, History), !.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The four place predicate provestoup attempts to
% prove a sequent with a history and a stoup
% formula, of form Context ->(Stoup) G ; History.
% Again we have first argument indexing.  The cut in the axiom
% rule prevents prolog from trying to redo the axioms.
% The predicate takes arguments
% provestoup(Stoup, Goal, Context, History).
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

provestoup(A, A, _, _):-!.

provestoup(falsum, _, _, _).

provestoup(A imp B, G, Context, History):-
          non_member(G, History), !,
          provestoup(B, G, Context, History), !,
          provenostoup(A, Context, [G | History]), !.

provestoup(not(A), G, Context, History):-
          non_member(G, History), !,
          provenostoup(A, Context, [G | History]), !.

provestoup(A and B, G, Context, History):-
          provestoup(A, G, Context, History), !.

provestoup(A and B, G, Context, History):-
          provestoup(B, G, Context, History), !.

provestoup(A or B, G, Context, _):-
          non_member(A, Context), !,
          non_member(B, Context), !,
          provenostoup(G, [A | Context], []), !,
          provenostoup(G, [B | Context], []), !.
```

4 Theorem Proving with a Scottish History

In this section we shall briefly discuss the second calculus presented. The calculus adds to the history at several points rather than just the one as in the case of the Swiss history, and so has to store a larger set. It also checks for looping more often than in the Swiss history, and so proof trees do not have to be so large.

4.1 Theory for the non-Swiss MJ^{Hist}

With the Swiss style history all of the action takes place at one point; in the $(\supset_{\mathcal{L}})$ rule. This has the advantage that loop checking only forms part of one rule, and so for other rules less work has to be done than might otherwise be the case. We have also noted that this style of history mechanism ensures that only a very small amount of information is stored in the history (consisting of only propositional variables, falsum and disjunctions, when condition $\star$ is included). However there is a tradeoff between these advantages and the obvious disadvantage of not looking for loops very often. We will find loops more quickly if we look for them at more points. That is, we might continue building a tree needlessly, when a loop might already have been spotted. This is the motivation for the alternative calculus. It has a larger history, but this allows us to check for loops more often, the idea being that in certain situations this might be an advantage. For example, the sequent:

$$p, q, (p \supset q \supset r) \supset r \Rightarrow p \supset q \supset r$$

will give the following in the Swiss style MJ^{Hist} (where $\Gamma = p, q, (p \supset q \supset r) \supset r$):

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma \Rightarrow p \supset q \supset r; \{r\} \quad \dfrac{}{\Gamma \xrightarrow{r} r; \varepsilon}\,(ax)}{\Gamma \xrightarrow{(p \supset q \supset r) \supset r} r; \varepsilon}\,(C)}{\Gamma \Rightarrow r; \varepsilon}\,(\supset_{\mathcal{R}})}{\Gamma \Rightarrow q \supset r; \varepsilon}\,(\supset_{\mathcal{R}})}{\Gamma \Rightarrow p \supset q \supset r; \varepsilon}\,(\supset_{\mathcal{R}})$$

and we have to go through all the inference steps again (in the branch above the left premiss) before the loop is detected. However, in the alternative calculus we get:

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma \Rightarrow p \supset q \supset r \quad \Gamma \xrightarrow{r} r; \{r, q \supset r, p \supset q \supset r\}}{\Gamma \xrightarrow{(p \supset q \supset r) \supset r} r; \{r, q \supset r, p \supset q \supset r\}}\,(\supset_{\mathcal{L}})}{\Gamma \Rightarrow r; \{r, q \supset r, p \supset q \supset r\}}\,(C)}{\Gamma \Rightarrow q \supset r; \{q \supset r, p \supset q \supset r\}}\,(\supset_{\mathcal{R}})}{\Gamma \Rightarrow p \supset q \supset r; \{p \supset q \supset r\}}\,(\supset_{\mathcal{R}})$$

The final inference, $(\supset_{\mathcal{L}})$, is not valid, due to the left premiss having a goal sequent, $p \supset q \supset r$, which is already in the history. That is, the loop is detected, and is detected more quickly than in the Swiss style calculus.

We may prove the two equivalence theorems for this second calculus, analogously to those for the Swiss-style calculus. The proofs work in the same way, although the detail is different. Details are omitted.

Theorem 3 *The calculi MJ and MJ^{Hist} (without $*$) are equivalent. That is, a sequent $\Gamma \Rightarrow A$ is provable in MJ if and only if $\Gamma \Rightarrow A; A$ (the sequent with its trivial history) is provable in MJ^{Hist} (without $*$).*

PROOF: Similar to proof of Theorem 1. ■

Theorem 4 *The calculus MJ^{Hist} with condition $*$ placed on rule (C) is equivalent to MJ^{Hist} without the extra condition.*

PROOF: Similar to proof of Theorem 2. ■

4.2 Implementation of the Scottish MJ^{Hist}

Again the Prolog implementation is much the same as for the Swiss calculus. All that is changed is that formulae are added to the history lists in the appropriate places and the extra history checks are added.

5 Proof Search in MJ^{Hist}

5.1 What are we Searching For?

A proof of a sequent in MJ is a completed proof tree in that calculus. Proof search is the process of looking for all different proofs of a certain sequent, where two proofs are different if they correspond to different normal natural deductions. As MJ is designed so that deductions in MJ are in 1-1 correspondence with normal natural deductions, the challenge is to enumerate all possible proof trees for a given root. As MJ^{Hist} is designed for theorem proving, and does not allow loops, it cannot possibly find all proofs, as many proofs contain loops. However, we can perform partial proof search, finding a subset of the proofs. Namely, all loop free proofs.

To perform proof search, we obviously do not want restriction $\star$, as this would prevent us finding many proofs. We also do not want the cuts in the implementation. Notice also that the two versions of MJ^{Hist} will find different proofs. The Swiss version has the delayed loop checking mentioned above. This allows some proofs with a single loop in them to be valid in the calculus. For example, consider the sequent:

$$\Rightarrow (p \supset p) \supset (p \supset p)$$

Both versions of the calculus (without $\star$) allow the following two proofs (with the relevant histories omitted):

$$\dfrac{\dfrac{\dfrac{}{p \supset p \xrightarrow{p \supset p} p \supset p}\,(ax)}{p \supset p \Rightarrow p \supset p}\,(C)}{\Rightarrow (p \supset p) \supset (p \supset p)}\,(\supset_{\mathcal{R}}) \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{p, p \supset p \xrightarrow{p} p}\,(ax)}{p, p \supset p \Rightarrow p}\,(C)}{p \supset p \Rightarrow p \supset p}\,(\supset_{\mathcal{R}})}{\Rightarrow (p \supset p) \supset (p \supset p)}\,(\supset_{\mathcal{R}})$$

But the Swiss-style history also allows this (but no other proofs):

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{p, p \supset p \xrightarrow{p} p; \{p\}}\,(ax)}{p, p \supset p \Rightarrow p; \{p\}}\,(C) \qquad \dfrac{}{p, p \supset p \xrightarrow{p} p; \varepsilon}\,(ax)}{p, p \supset p \xrightarrow{p \supset p} p; \varepsilon}\,(\supset_{\mathcal{L}})}{p, p \supset p \Rightarrow p; \varepsilon}\,(C)}{p \supset p \Rightarrow p \supset p; \varepsilon}\,(\supset_{\mathcal{R}})}{\Rightarrow (p \supset p) \supset (p \supset p); \varepsilon}\,(\supset_{\mathcal{R}})$$

and all the proofs in MJ are described by:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\begin{matrix}\vdots\\ P, P \supset P \Rightarrow P\end{matrix} \quad \dfrac{}{P, P \supset P \xrightarrow{P} P}\,(ax)}{P, P \supset P \xrightarrow{P \supset P} P}\,(\supset_{\mathcal{L}})}{P, P \supset P \Rightarrow P}\,(C) \qquad \dfrac{}{P, P \supset P \xrightarrow{P} P}\,(ax)}{P, P \supset P \xrightarrow{P \supset P} P}\,(\supset_{\mathcal{L}})}{P, P \supset P \Rightarrow P}\,(C)}{P \supset P \Rightarrow P \supset P}\,(\supset_{\mathcal{R}})}{\Rightarrow (P \supset P) \supset (P \supset P)}\,(\supset_{\mathcal{R}})$$

plus the first proof given for the non-Swiss history.

5.2 Implementation

In this section we give an implementation of partial proof search using MJ^{Hist}. The code presented is for the Scottish calculus, the code for the Swiss calculus is similar. The main difference in the code (apart from the addition of arguments to save and print out proofs) is that the cuts are removed from the (ax) rule. This is so that in backtracking these can be undone and different proofs found. The other main difference is that the contraction rule is now one single general rule rather than three specific rules, so that now any formula can be placed in the stoup, again increasing the number of proofs that can be found. That is, we don't have restriction $\star$.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% We need use_module(library(lists)) for nth, member
% and non_member.
%
% The two place predicate show is used to
% display the proofs (see results).
% 's' is used as a marker, to indicate
% when the proof tree splits
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

show([], N).

show([s | X], N):-!,
          nth(1, X, A),
          nth(2, X, B),
          nth(3, X, C),
          nth(4, X, D),
          show([A | B], N),
          show([C | D], N).

show([X | Y], N):-!,
          ttytab(N),
          write(X), nl,
          NewN is (N+3),
          show(Y, NewN).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% proofsearch finds a proof, displays it and
% then uses fail to make prolog backtrack to
% find another proof.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

proofsearch(G, Context):-
          prove(G, Context, Tree),
          show(Tree, 0), nl, nl,
          fail.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% From hereon, the code is much the same as for
% the theorem provers, with the addition of an
% extra argument which stores the proof so far
% as a list of ordered triples and quadruples.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

prove(G, Context, [(G, Context, [G]) | Tree]):-
          provenostoup(G, Context, [G], Tree).

provenostoup(A imp B, Context, _, [(B, [A | Context],
                   [B]) | Tree]):-
          non_member(A, Context),
          provenostoup(B, [A | Context], [B], Tree).

provenostoup(A imp B, Context, History, [(B, Context,
                   [B | History]) | Tree]):-
          member(A, Context),
          non_member(B, History),
          provenostoup(B, Context, [B | History], Tree).

provenostoup(not(A), Context, History, [(falsum,
               [A | Context], [falsum]) | Tree]):-
          non_member(A, Context),
          provenostoup(falsum, [A | Context], [falsum],
                   Tree).

provenostoup(not(A), Context, History, [(falsum, Context,
                [falsum | History]) | Tree]):-
           member(A, Context),
           non_member(falsum, History),
           provenostoup(falsum, Context, [falsum | History],
                     Tree).

provenostoup(A and B, Context, History, [s, (A, Context,
[A | History]), Tree1, (B, Context, [B | History]), Tree2]):-
           non_member(A, History),
           non_member(B, History),
           provenostoup(A, Context, [A | History], Tree1),
           provenostoup(B, Context, [B | History], Tree2).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The use of a variable as the first argument of member
% allows any formula to be selected by member, and on
% backtracking the next member is selected.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

provenostoup(A, Context, History, [(X, A, Context, History)
                     | Tree]):-
           member(X, Context),
           provestoup(X, A, Context, History, Tree).

provenostoup(A or B, Context, History, [(A, Context,
                     [A | History]) | Tree]):-
           non_member(A, History),
           provenostoup(A, Context, [A | History], Tree).

provenostoup(A or B, Context, History, [(B, Context,
                [B | History]) | Tree]):-
           non_member(B, History),
           provenostoup(B, Context, [B | History], Tree).

provestoup(A, A, _, _, []).

provestoup(falsum, _, _, _, []).

provestoup(A imp B, G, Context, History, [s, (B, G, Context,
      History), Tree1, (A, Context, [A | History]), Tree2]):-
           non_member(A, History),
           provestoup(B, G, Context, History, Tree1),
           provenostoup(A, Context, [A | History], Tree2).

provestoup(not(A), G, Context, History, [(A, Context,
                     [A | History]) | Tree]):-
           non_member(A, History),
          provenostoup(A, Context, [A | History], Tree).

provestoup(A and B, G, Context, History, [(A, G, Context,
                    History) | Tree]):-
          provestoup(A, G, Context, History, Tree).

provestoup(A and B, G, Context, History, [(B, G, Context,
                    History) | Tree]):-
          provestoup(B, G, Context, History, Tree).

provestoup(A or B, G, Context, _, [s, (G, [A | Context], [G]),
      Tree1, (G, [B | Context], [G]), Tree2]):-
          non_member(A, Context),
          non_member(B, Context),
          provenostoup(G, [A | Context], [G], Tree1),
          provenostoup(G, [B | Context], [G], Tree2).
```

6 Results

6.1 Theorem Proving

The issue we are concerned with here is that of speed, how quickly we find out whether or not a certain sequent or formula is provable. We tested the two theorem provers on a sample of problems, some easy, some more problematic.

The programs were run using SICStus Prolog2.1 on a SUN SparcStation 10. The times given are runtimes (in milliseconds), i.e. ”CPU time used whilst executing, excluding time spent garbage collecting, stack shifting or in system calls” [7]. The times are averages. We first give the formula represented with the normal symbols, then as it is coded for the the program. Next we give a result of provable/unprovable, followed by times for two calculi.

Example 1.

$((A \vee B \vee C) \wedge (D \vee E \vee F) \wedge (G \vee H \vee J) \wedge (K \vee L \vee M)) \supset$
$(A \wedge D) \vee (A \wedge G) \vee (A \wedge K) \vee (D \wedge G) \vee (D \wedge K) \vee (G \wedge K) \vee$
$(B \wedge E) \vee (B \wedge H) \vee (B \wedge L) \vee (E \wedge H) \vee (E \wedge L) \vee (H \wedge L) \vee$
$(C \wedge F) \vee (C \wedge J) \vee (C \wedge M) \vee (F \wedge J) \vee (F \wedge M) \vee (J \wedge M)$

```
((pv(a) or pv(b) or pv(c)) and (pv(d) or pv(e) or pv(f)) and
(pv(g) or pv(h) or pv(j)) and (pv(k) or pv(l) or pv(m))) imp
((pv(a) and pv(d)) or (pv(a) and pv(g)) or (pv(a) and pv(k))
or (pv(d) and pv(g)) or (pv(d) and pv(k)) or (pv(g) and pv(k))
or (pv(b) and pv(e)) or (pv(b) and pv(h)) or (pv(b) and pv(l))
or (pv(e) and pv(h)) or (pv(e) and pv(l)) or (pv(h) and pv(l))
or (pv(c) and pv(f)) or (pv(c) and pv(j)) or (pv(c) and pv(m))
or (pv(f) and pv(j)) or (pv(f) and pv(m)) or (pv(j) and pv(m)))
provable
Swiss history 1388
Scottish 1701
```

Example 2.

$$((A \vee B) \wedge (D \vee E) \wedge (G \vee H))$$
$$\supset (A \wedge D) \vee (A \wedge G) \vee (D \wedge G) \vee (B \wedge E) \vee (B \wedge H) \vee (E \wedge H)$$

```
((pv(a) or pv(b) or pv(c)) and (pv(d) or pv(e) or pv(f))) imp
((pv(a) and pv(d)) or (pv(b) and pv(e)) or (pv(c) and pv(f)))
unprovable
Swiss history 15
Scottish 21
```

Example 3.

$$(A \supset B) \supset (A \supset C) \supset (A \supset (B \wedge C))$$

```
(pv(a) imp pv(b)) imp (pv(a) imp pv(c)) imp (pv(a) imp (pv(b)
and pv(c)))
provable
Swiss history 0.2
Scottish 0.2
```

Example 4.

$$(A \wedge \neg A) \supset B$$

```
(pv(a) and not(pv(a))) imp pv(b)
provable
Swiss history 0.1
Scottish 0.1
```

Example 5.

$$(A \vee C) \supset (A \supset B) \supset (B \vee C)$$

```
(pv(a) or pv(c)) imp (pv(a) imp pv(b)) imp (pv(b) or pv(c))
provable
Swiss history 0.6
Scottish 0.8
```

Example 6.

$$((((A \supset B) \wedge (B \supset A)) \supset (A \wedge B \wedge C)) \wedge (((B \supset C) \wedge (C \supset B)) \supset (A \wedge B \wedge C)) \wedge$$
$$(((C \supset A) \wedge (A \supset C)) \supset (A \wedge B \wedge C))) \supset (A \wedge B \wedge C)$$

```
((((pv(a) imp pv(b)) and (pv(b) imp pv(a))) imp (pv(a) and
pv(b) and pv(c))) and (((pv(b) imp pv(c)) and (pv(c) imp
pv(b))) imp (pv(a) and pv(b) and pv(c))) and (((pv(c) imp
pv(a)) and (pv(a) imp pv(c))) imp (pv(a) and pv(b) and pv(c))))
imp (pv(a) and pv(b) and pv(c))
```

```
provable
Swiss history 11
SCottish 14
```

Example 7.
$((\neg\neg P \supset P) \supset P) \vee (\neg P \supset \neg P) \vee (\neg\neg P \supset \neg\neg P) \vee (\neg\neg P \supset P)$

```
((not(not(pv(p))) imp pv(p)) imp pv(p)) or (not(pv(p)) imp
not(pv(p))) or (not(not(pv(p))) imp not(not(pv(p)))) or
(not(not(pv(p))) imp pv(p))
provable
Swiss history 0.5
Scottish 0.5
```

Example 8.
$(((G \supset A) \supset J) \supset D \supset E) \supset$
$(((H \supset B) \supset I) \supset C \supset J \supset (A \supset H) \supset F \supset G \supset$
$(((C \supset B) \supset I) \supset D) \supset (A \supset C) \supset (((F \supset A) \supset B) \supset I) \supset E)$

```
(((pv(g) imp pv(a)) imp pv(j)) imp pv(d) imp pv(e)) imp
(((pv(h) imp pv(b)) imp pv(i)) imp pv(c) imp pv(j) imp (pv(a)
imp pv(h)) imp pv(f) imp pv(g) imp (((pv(c) imp pv(b)) imp
pv(i)) imp pv(d)) imp (pv(a) imp pv(c)) imp (((pv(f) imp
pv(a)) imp pv(b)) imp pv(i)) imp pv(e))
provable
Swiss history 2.2
Scottish 2.6
```

Example 9.
$A \supset B \supset ((A \supset B \supset C) \supset C) \supset (A \supset B \supset C)$

```
pv(a) imp pv(b) imp ((pv(a) imp pv(b) imp pv(c)) imp pv(c))
imp (pv(a) imp pv(b) imp pv(c))
unprovable
Swiss history 0.4
Scottish 0.5
```

Example 10.
$\neg\exists y \forall x(f(x,y) \leftrightarrow \neg\exists(f(x,z) \wedge f(z,x)))$
Instantiated over a two element universe.

```
provable
Swiss history 10082
Scottish 47
```

6.2 Proof Search

In this section we will give some examples of the results returned when proof search is performed. The program was run with the same software and hardware as in the previous section.

Given the number of different proofs that exist for any complicated formula, all we shall do is demonstrate the output for a couple of very simple examples.

The layout is as follows. The sequents are given as order triples or quadruples, depending in on whether they are stouped or unstouped. The unstouped sequents are given as triples (Goal, Context, History), the stouped sequents are given as quadruples (Stoup, Goal, Context, History). The proof trees are given upside down. That is, the first sequent given is the root. A half centimetre indent indicates that a sequent is above the previous one. Left branches are written first.

Example1.
$(P \supset P) \supset (P \supset P)$
```
(pv(p) imp pv(p)) imp (pv(p) imp pv(p))
```
Scottish output
```
(pv(p) imp pv(p)) imp (pv(p) imp pv(p)), [], [(pv(p) imp pv(p)
                         imp (pv(p) imp pv(p))]
    pv(p) imp pv(p), [pv(p) imp pv(p)], [pv(p) imp pv(p)]
       pv(p), [pv(p), pv(p) imp pv(p)], [pv(p)]
          pv(p), pv(p), [pv(p), pv(p) imp pv(p)], [pv(p)]

(pv(p) imp pv(p)) imp (pv(p) imp pv(p)), [], [(pv(p) imp pv(p)
                         imp (pv(p) imp pv(p))]
    pv(p) imp pv(p), [pv(p) imp pv(p)], [pv(p) imp pv(p)]
       pv(p) imp pv(p), pv(p) imp pv(p), [pv(p) imp pv(p)],
                         [pv(p) imp pv(p)]
```

Swiss output
```
(pv(p) imp pv(p)) imp (pv(p) imp pv(p)), [], []
    pv(p) imp pv(p), [pv(p) imp pv(p)], []
       pv(p), [pv(p), pv(p) imp pv(p)], []
          pv(p), pv(p), [pv(p), pv(p) imp pv(p)], []

(pv(p) imp pv(p)) imp (pv(p) imp pv(p)), [], []
    pv(p) imp pv(p), [pv(p) imp pv(p)], []
       pv(p) imp pv(p), pv(p) imp pv(p), [pv(p) imp pv(p)], []

(pv(p) imp pv(p)) imp (pv(p) imp pv(p)), [], []
    pv(p) imp pv(p), [pv(p) imp pv(p)], []
       pv(p), [pv(p), pv(p) imp pv(p)], []
          pv(p) imp pv(p), pv(p), [pv(p), pv(p) imp pv(p)], []
             pv(p), pv(p), [pv(p), pv(p) imp pv(p)], []
```

```
            pv(p), [pv(p), pv(p) imp pv(p)], [pv(p)]
              pv(p), pv(p), [pv(p), pv(p) imp pv(p)], [pv(p)]
```

Example2.
$(A \vee B) \supset (B \vee A)$

```
(pv(a) or pv(b)) imp (pv(b) or pv(a))
```

Swiss proof

```
(pv(a) or pv(b)) imp (pv(b) or pv(a)), [], []
  pv(b) or pv(a), [pv(a) or pv(b)], []
    pv(a) or pv(b), pv(b) or pv(a), [pv(a) or pv(b)], []
      pv(b) or pv(a), [pv(a), pv(a) or pv(b)], []
        pv(a), [pv(a), pv(a) or pv(b)], []
          pv(a), pv(a), [pv(a), pv(a) or pv(b)], []
      pv(b) or pv(a), [pv(b), pv(a) or pv(b)], []
        pv(b), [pv(b), pv(a) or pv(b)], []
          pv(b), pv(b), [pv(b), pv(a) or pv(b)], []
```

There is only one Scottish proof, which is the same as above with the appropriate changes made to the history.

7 Conclusion

The use of a pared down history makes for seemingly efficient loop detection. However as other intuitionistic theorem provers (and there are not that many) are written in different languages, are run on different machines and (in most cases) deal with first order formulae, comparison is hard. Of the two implementations given here, the one with the smallest history and the least checking (the Swiss one) can become inefficient (see example 10) when delay in loop checking compared with the other version allows many extra branches to be pursued. The inefficiency of the increased history is more than counterbalanced by the extra loop checks. We should point out that the times here compare badly with the implementation of the contraction-free LJT.

For proof search neither calculus is really suited, as they only find loop free proofs (plus a few more in the Swiss case), but if that is all you are after then these calculi will suit your purposes well.

References

[1] Dyckhoff, R. and Pinto, L.: *A Permutation-free Sequent Calculus for Intuitionistic Logic*. unpublished (1996)

[2] Dyckhoff, R. and Pinto, L.: *Permutability of Proofs in Intuitionistic Sequent Calculi*. unpublished (1996)

[3] Gabbay, D.: *Algorithmic Proof With Diminishing Resources, part 1*. LNCS 533 (1991) pp. 156-173

[4] Girard, J.-Y.: *A New Constructive Logic: Classical Logic*. Mathematical Structures in Computer Science, 1991(1), pp. 255-296

[5] Herbelin, H.: *A λ-calculus Structure Isomorphic to Gentzen-style Sequent Calculus Structure*. LNCS 933 (1995) pp. 61-75

[6] Heuerding, A., Seyfried, M., Zimmermann, H.: *Efficient Loop-Check for Backward Proof Search in Some Non-classical Propositional Logics*. LNCS 1071 (1996) pp. 210-225

[7] *SICStus Prolog User's Manual*. Swedish Institute of Computer Science (1993)