Two Loop Detection Mechanisms: a Comparison

Jacob M. Howe

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

Abstract. In order to compare two loop detection mechanisms we describe two calculi for theorem proving in intuitionistic propositional logic. We call them both MJ^{Hist}, and distinguish between them by description as 'Swiss' or 'Scottish'. These calculi combine in different ways the ideas on focused proof search of Herbelin and Dyckhoff & Pinto with the work of Heuerding *et al* on loop detection. The Scottish calculus detects loops earlier than the Swiss calculus but at the expense of modest extra storage in the history. A comparison of the two approaches is then given, both on a theoretic and on an implementational level.

1 Introduction

The main interest of this paper is the comparison of the two loop detection mechanisms described below. In order to do this we illustrate their use on the permutation-free sequent calculus MJ for the propositional fragment of intuitionistic logic. This gives calculi whose implementations are suitable for theorem proving.

Backwards proof search and theorem proving with a standard cut-free sequent calculus, Gentzen's LJ, for the propositional fragment of intuitionistic logic is inefficient because of three problems. 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 and where to apply them.

The sequent calculus MJ for intuitionistic logic was introduced (with another name, LJT) by Herbelin in [7]. The propositional fragment of the calculus MJ is displayed in Figure 1. This uses Girard's idea of a special place for formulae in the antecedent, the stoup first seen in [6]. The calculus was developed by Dyckhoff and Pinto [3] because it has the property that proofs are in 1–1 correspondence with the normal natural deductions of NJ. MJ is a permutation-free sequent calculus; it avoids the problems of permutations in the cut-free sequent calculus of Gentzen. This removes the second of the problems. In this paper we are more interested in theorem proving than in proof search, hence the second problem is not directly relevant. But notice that permutations are avoided in MJ by a focusing method — several choice points are removed. That is, MJ partly addresses the third problem and hence is advantageous as a calculus for theorem proving. However, the naïve implementation of MJ will lead to the possibility of looping.

Looping may easily be removed by checking whether a sequent has already occurred in a branch. Implementation of this is inefficient as it requires much information to be stored. Recent work by Heuerding *et al* [9] (the intuitionistic case of which is closely related to that of Gabbay in [5]) 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 the resulting calculi have advantages and disadvantages. These are discussed theoretically and also pragmatically (in terms of the speed with which Prolog implementations give proofs). We call the new calculus MJ^{Hist}, the two varieties 'Swiss' and 'Scottish'.

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

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

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

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

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

Define $\neg A \equiv A \supset \perp$.
A, B, C are formulae. Γ is a multiset of formulae.
B, Γ is shorthand for $\{B\} \cup \Gamma$, where $\cup$ is multiset union.
Sequent $\Gamma \Rightarrow C$ has context Γ, goal C and no stoup.
Sequent $\Gamma \xrightarrow{A} C$ has context Γ, goal C and a single formula, A, in the stoup.

Fig. 1. Propositional Fragment of the calculus MJ.

2 Calculi With Histories

In this section we first discuss the idea of the history mechanism, and then describe the two calculi. We shall conclude with a comparison of the two calculi.

2.1 The Use of Histories to Prevent Looping

Looping can very easily occur in MJ, for example:

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

The sequent $(p \wedge p) \supset p \Rightarrow p$ may continue to occur in the proof tree for this sequent using the MJ calculus. We can see that there is a loop: we need a mechanical way to detect 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 on the branch of 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 backtrack. If not the new history is the extension of the old history by the old sequent (without its history), and we try to prove the new sequent, and so on. Unfortunately this method is 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. Efficiency would be improved by cutting down the amount of storage and checks to the bare minimum needed to prevent looping.

The basis of the reduced history is the realisation (as in [9]) 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 be 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. Therefore we may empty the history every time the context is extended. All we need store in the history are goal formulae. If we have a sequent whose goal is already in the history, then we have the same goal and the same context as another sequent, that is, a loop.

There are two slightly different ways of doing this. There is the straightforward extension and modification of the calculus described in [9] (which we call a 'Swiss history'). The other approach involves storing slightly more formulae in the history, but detects loops more quickly. This we describe as the 'Scottish history'; it can, in many cases, be much more efficient than the Swiss method.

2.2 The Swiss History

Before continuing, we should point out that the calculus we describe here as Swiss is significantly different from the one in [9]. This is partly due to our use of MJ as a base calculus, and partly because we are trying to focus on the history mechanism, hence we have not included the subsumption checks that the calculus in [9] uses.

The Swiss-style calculus MJ^{Hist} is displayed in Figure 2. Let us make some general points about it (which will apply to the Scottish MJ^{Hist} as well). We have given explicit rules for negation (which are just special cases of the rules for implication) for the sake of completeness of connectives. Also, notice that there are two rules for $(\supset_{\mathcal{R}})$.

These correspond to the two cases where the new formula is or is not in the context. As noted above (in §2.1) this is very important for MJ^{Hist}. Also note that the number of formulae in the history is at most equal to the length of the formula we check for provability.

The loop checking due to the history in the calculus works in a similar way to that of $\mathsf{IPC}^{RP}{}_{\wedge,\rightarrow}{}^{SU}$ in [9]. A sequent is matched against first the conclusions of right rules until the goal formula is either a propositional variable, falsum, or a disjunction (note that disjunction isn't covered in [9], and requires special treatment). This is ensured by condition * on rule (C). Then a formula from the context is selected and placed in the stoup by the (C) rule, the sequent is then matched against stoup formulae of left rules (this focusing does not occur in [9]). The history mechanism is used to prevent looping in the $(\supset_{\mathcal{L}})$ rule (and similarly in the $(\neg_{\mathcal{L}})$ rule). The left premiss of the rule has the same context as the conclusion, but the goal is generally different. If the goal, C, of the conclusion is not in the history, $\mathcal{H}$, we store C in $\mathcal{H}$ and continue backwards proof search on the left premiss. Alternatively, C might already be in $\mathcal{H}$. In this case there is a loop, and so this branch is not pursued. We backtrack and look for a proof in a different way.

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 when it is not extended. Something similar is happening in 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, S, will be the same as some sequent at a lesser height in the proof tree — there is a loop. This is easy to see: since the context and goal of S 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 S.

We now state the equivalence theorem. This is done in two stages.

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

PROOF:(Sketch) The $\Leftarrow$ direction is straightforward.

To prove the $\Rightarrow$ direction we take an MJ proof tree and use it to build an MJ^{Hist} proof tree.

We start at the root, $\Gamma \Rightarrow A$ in MJ and we have root $\Gamma \Rightarrow A; \{A\}$ in MJ^{Hist}.

Given a fragment of MJ proof tree with corresponding fragment of MJ^{Hist} proof tree, we look at the next inference in the MJ tree. We have a recipe which we can use to build a fragment of MJ^{Hist} proof tree corresponding to a strictly larger fragment of the MJ proof tree.

As proof trees are finite, this process must be terminating.

For full details see [10]. ■

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

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

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

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

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

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

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

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

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

$\star$ B is either a propositional variable, $\bot$ or a disjunction.
A, B, C are formulae, Γ is a multiset of formulae, and $\mathcal{H}$ is a set of formulae.
B, Γ is shorthand for $\{B\} \cup \Gamma$.
Sequent $\Gamma \Rightarrow C; \mathcal{H}$ has context Γ, goal C, history $\mathcal{H}$ and no stoup.
Sequent $\Gamma \xrightarrow{A} C; \mathcal{H}$ has context Γ, goal C, history $\mathcal{H}$ and stoup A.
When the history has been extended we have parenthesised $(C, \mathcal{H})$ for emphasis.

Fig. 2. The propositional calculus MJ^{Hist}, Swiss style

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

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

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

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

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

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

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

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

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

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

$\star$ B is either a propositional variable, $\bot$ or a disjunction.
A, B, C are formulae, Γ is a multiset of formulae, $\mathcal{H}$ is a set of formulae.
B, Γ is shorthand for $\{B\} \cup \Gamma$.
Sequent $\Gamma \Rightarrow C; \mathcal{H}$ has context Γ, goal C, history $\mathcal{H}$ and no stoup.
Sequent $\Gamma \xrightarrow{A} C; \mathcal{H}$ has context Γ, goal C, history $\mathcal{H}$ and stoup A.
When the history has been extended we have parenthesised $(C, \mathcal{H})$ for emphasis.

Fig. 3. The propositional calculus MJ^{Hist}, Scottish style

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

PROOF:(Sketch) The $\Leftarrow$ direction is trivial.

To prove the $\Rightarrow$ direction, we first prove that MJ and MJ with condition $\star$ on (C) are equivalent. This is done by a simple induction on the depth of the proof and on complexity of formulae.

For any MJ^{Hist} (without $\star$) proof that doesn't satisfy $\star$, we can consider it as an MJ proof. Then we can find an MJ proof satisfying $\star$. Using the procedure in the proof of theorem 1, we can build an MJ^{Hist} (with $\star$) proof tree.

For full details see [10] ■

2.3 The Scottish History

In this section we discuss the Scottish MJ^{Hist}. We go through its theory where it is different from the Swiss style calculus and explaining the motivations for the alternative approach. The Scottish MJ^{Hist} is given in Figure 3.

We said earlier that when using a history mechanism to prevent looping it would be good to cut down the amount of storage and checking needed to a bare minimum. This was done in the Swiss MJ^{Hist} — the history mechanism operates in one place only and other restrictions for loop prevention involve no storage. However it is not clear that this is the best and most attractive approach. 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. The Scottish MJ^{Hist} has larger histories, but this allows us to check for loops more often, and in certain situations this is advantageous.

As in the Swiss history, when attempting to prove a sequent, right rules are applied first, then (C), then left rules. Also, looping is prevented by the $(\vee_{\mathcal{L}})$ rule in the same way. The difference between the two calculi is in the way that the history mechanism works.

Whereas the Swiss calculus only places formulae in the history which have been the goal of the conclusion of a $(\supset_{\mathcal{L}})$ (or $(\neg_{\mathcal{L}})$) rule, the Scottish calculus keeps as the history a complete record of the goal formulae of sequents between context extensions. At each of the places where the history might be extended, the new goal is checked against the history. If it is in the history, then there is a loop. The heart of the difference between the two calculi is that in the Swiss calculus loop checking is done when a formula leaves the goal, whereas in the Scottish calculus it is done when it becomes the goal.

We have the same equivalence theorems as for the Swiss calculus. These are proved in a similar manner. For details again see [10].

Theorem 3 *The calculi MJ and MJ^{Hist} (without $\star$) 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 $\star$).*

PROOF: Similar to that of theorem 1. ■

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

PROOF: Similar to that of theorem 2. ■

2.4 Comparison of the Two Calculi

Because of the way that the Swiss history works, loop detection is delayed. Let us illustrate this with an example. Consider the sequent:

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

In the Swiss style MJ^{Hist} (where $\Gamma = p, q, (p \supset q \supset r) \supset r$, and $G = p \supset q \supset r$) this gives the following:

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma \Rightarrow G; \{r\} \quad \dfrac{}{\Gamma \xrightarrow{r} r; \phi}(ax)}{\Gamma \xrightarrow{(p\supset q\supset r)\supset r} r; \phi}(\supset_{\mathcal{L}})}{\Gamma \Rightarrow r; \phi}(C)}{\Gamma \Rightarrow q \supset r; \phi}(\supset_{\mathcal{R}2})}{\Gamma \Rightarrow G; \phi}(\supset_{\mathcal{R}2})$$

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 Scottish calculus we get:

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma \Rightarrow G; \{G, r, q \supset r, G\} \quad \dfrac{}{\Gamma \xrightarrow{r} r; \{r, q \supset r, G\}}(ax)}{\Gamma \xrightarrow{(p\supset q\supset r)\supset r} r; \{r, q \supset r, G\}}(\supset_{\mathcal{L}})}{\Gamma \Rightarrow r; \{r, q \supset r, G\}}(C)}{\Gamma \Rightarrow q \supset r; \{q \supset r, G\}}(\supset_{\mathcal{R}2})}{\Gamma \Rightarrow G; \{G\}}(\supset_{\mathcal{R}2})$$

The topmost inference, $(\supset_{\mathcal{L}})$, is not valid, because the left premiss has goal formula, G, which is already in the history. That is, the loop is detected, and is detected lower in the proof tree than in the Swiss style calculus.

Spotting the loop as it occurs is not only theoretically more attractive, but could also prevent a lot of costly extra computation.

The two calculi both have their good points. The Swiss calculus is efficient from the point of view that its history mechanism requires little storage and checking. The Scottish calculus is efficient in that it detects loops as they occur, avoiding unnecessary computations.

The question is whether or not in general an overhead in storage and checking of the history (which shouldn't be too great due to regular resetting) is preferable to the larger proof trees which are the result delaying checking. Perhaps the best way to decide this is to look at empirical results in the form of timings for implementations of the calculi. Note that as the two calculi are rather similar it is more than likely that any optimisation that can be applied to the one can be applied to the other.

3 Implementation of the Decision Procedure

Our implementation of the calculus is syntax directed. A sequent $\Gamma \Rightarrow A; \phi$ for the Swiss calculus, or $\Gamma \Rightarrow A; \{A\}$ for the Scottish, is passed to the theorem prover. For a sequent with an empty stoup, the next inference is determined by the goal. If the goal is an implication, negation or conjunction, then the appropriate rule on the right is applied. If an instance of one of these rules fails, then we have to backtrack as no other rule is applicable. If the goal is a propositional variable, falsum or a disjunction, the contraction rule is applied, selecting a formula and placing it in the stoup. If a contraction fails, then a different formula is placed in the stoup. If the goal is a propositional variable or falsum, and contraction has failed for all possible stoup formulae, we backtrack. If the goal is a disjunction and contraction has failed for all possible stoup formulae, then we may apply disjunction on the right. If this fails we have to backtrack. For a sequent with a stoup formula, the next inference is decided upon by the stoup formula. The next inference must be an instance of the appropriate rule on the left. If such an inference fails, then we have to backtrack. Note that in $(\supset_{\mathcal{L}})$ we check the right branch, the one with the stoup formula, first. We get failure if at any point no rule instance can be applied. We give an example of failure due to the history:

$$\frac{}{p, \Gamma \Rightarrow p \supset q; \{p, q\}} \; (\supset_{\mathcal{R}2})$$

fails due to $q \notin \{p, q\}$ not being satisfied. Because of condition $\star$, no other rule instances are applicable to this sequent and so we must backtrack.

For this implementation we do not need to know anything about the invertibility of any of the rules. However, it may be of some independent interest to point out rules which are invertible and those which are not. For all three calculi - MJ, MJ^{Hist} (Swiss) and MJ^{Hist} (Scottish) - all rules are fully invertible with the exception of $(\wedge_{\mathcal{L}})$, $(\vee_{\mathcal{R}})$ and (C).

4 Results

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 calculi were implemented in prolog (naïvely, code can be found in [10]). 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" [15]. In Figure 4 we present the formulae we gave to the theorem prover (the quantified formulae were instantiated over finite universes). In Table 1 we give the results and average timings (where NR means that the machine had not proved the example after running overnight).

The results indicate that although the Swiss calculus can be quicker on some examples, this advantage is less significant than the disadvantage of the several examples where the Swiss calculus is several orders of magnitude slower than the Scottish calculus. It should also be added that the times for the calculi implemented compare poorly with our implementation of the single succedant LJT calculus of [1].

1. $((A \vee B) \wedge (D \vee E) \wedge (G \vee H)) \supset ((A \wedge D) \vee (A \wedge G) \vee (D \wedge G) \vee (B \wedge E) \vee (B \wedge H) \vee (E \wedge H))$
2. $((A \vee B \vee C) \wedge (D \vee E \vee F) \wedge (G \vee H \vee J) \wedge (K \vee L \vee M)) \supset (A \wedge D) \vee (A \wedge G) \vee (A \wedge K) \vee (D \wedge G) \vee (D \wedge K) \vee (G \wedge K) \vee (B \wedge E) \vee (B \wedge H) \vee (B \wedge L) \vee (E \wedge H) \vee (E \wedge L) \vee (H \wedge L) \vee (C \wedge F) \vee (C \wedge J) \vee (C \wedge M) \vee (F \wedge J) \vee (F \wedge M) \vee (J \wedge M)$
3. $((A \vee B \vee C) \wedge (D \vee E \vee F)) \supset ((A \wedge B) \vee (B \wedge E) \vee (C \wedge F))$
4. $(A \supset B) \supset (A \supset C) \supset (A \supset (B \wedge C))$
5. $(A \wedge \neg A) \supset B$
6. $(A \vee C) \supset (A \supset B) \supset (B \vee C)$
7. $((((A \supset B) \wedge (B \supset A)) \supset (A \wedge B \wedge C)) \wedge (((B \supset C) \wedge (C \supset B)) \supset (A \wedge B \wedge C)) \wedge (((C \supset A) \wedge (A \supset C)) \supset (A \wedge B \wedge C))) \supset (A \wedge B \wedge C)$
8. $((\neg\neg P \supset P) \supset P) \vee (\neg P \supset \neg P) \vee (\neg\neg P \supset \neg\neg P) \vee (\neg\neg P \supset P)$
9. $(((G \supset A) \supset J) \supset D \supset E) \supset (((H \supset B) \supset I) \supset C \supset J) \supset (A \supset H) \supset F \supset G \supset (((C \supset B) \supset I) \supset D) \supset (A \supset C) \supset (((F \supset A) \supset B) \supset I) \supset E$
10. $A \supset B \supset ((A \supset B \supset C) \supset C) \supset (A \supset B \supset C)$
11. $((\neg\neg(\neg A \vee \neg B) \supset (\neg A \vee \neg B)) \supset (\neg\neg(\neg A \vee \neg B) \vee \neg(\neg A \vee \neg B))) \supset (\neg\neg(\neg A \vee \neg B) \vee \neg(\neg A \vee \neg B))$
12. $B \supset (A \supset (((A \wedge B) \supset C_1) \supset (((A \wedge B) \supset C_2) \supset (((A \wedge B) \supset C_3) \supset (((A \wedge B) \supset (B \supset C_1 \supset C_2 \supset C_3 \supset B)) \supset (A \wedge B))))))$
13. $((A \wedge B \vee C) \supset (C \vee (C \wedge D))) \supset (\neg A \vee ((A \vee B) \supset C))$
14. $\neg\neg((\neg A \supset B) \supset (\neg A \supset \neg B) \supset A)$
15. $\neg\neg(((A \leftrightarrow B) \leftrightarrow C) \leftrightarrow (A \leftrightarrow (B \leftrightarrow C)))$
16. $\forall x \exists y \forall z (p(x) \wedge q(y) \wedge r(z)) \leftrightarrow \forall z \exists y \forall x (p(x) \wedge q(y) \wedge r(z))$
17. $\exists x_1 \forall y_1 \exists x_2 \forall y_2 \exists x_3 \forall y_3 (p(x_1, y_1) \wedge q(x_2, y_2) \wedge r(x_3, y_3)) \supset \forall y_3 \exists x_3 \forall y_2 \exists x_2 \forall y_1 \exists x_1 (p(x_1, y_1) \wedge q(x_2, y_2) \wedge r(x_3, y_3))$
18. $\neg \exists x \forall y (mem(y, x) \leftrightarrow \neg mem(x, x))$
19. $\neg \exists x \forall y (q(y) \supset r(x, y)) \wedge \exists x \forall y (s(y) \supset r(x, y)) \supset \neg \forall x (q(x) \supset s(x))$
20. $\forall z_1 \forall z_2 \forall z_3 (q(z_1, z_2, z_3, z_1, z_2, z_3)) \supset \exists x_1 \exists x_2 \exists x_3 \exists y_1 \exists y_2 \exists y_3 ((p(x_1) \wedge p(x_2) \wedge p(x_3) \leftrightarrow p(y_1) \wedge p(y_2) \wedge p(y_3)) \wedge q(x_1, x_2, x_3, y_1, y_2, y_3))$
21. $((\exists x (p \supset f(x))) \wedge (\exists x_1 (f(x_1) \supset p))) \supset (\exists x_2 ((p \supset f(x_2)) \wedge (f(x_2) \supset p)))$
22. $(\exists x (p(x)) \wedge (\forall x_1 (f(x_1) \supset (\neg g(x_1) \wedge r(x_1)))) \wedge (\forall x_2 (p(x_2) \supset (g(x_2) \wedge f(x_2))) \wedge (\forall x_3 (p(x_3) \supset q(x_3)) \vee \exists x_4 (p(x_4) \wedge r(x_4)))))) \supset \exists x_5 (q(x_5) \wedge p(x_5))$
23. $((\exists x (p(x)) \leftrightarrow \exists x_1 (q(x_1))) \wedge \forall x_2 \forall y ((p(x_2) \wedge q(y)) \supset (r(x_2) \leftrightarrow s(y)))) \supset (\forall x_3 (p(x_3) \supset r(x_3)) \leftrightarrow \forall x_4 (q(x_4) \supset s(x_4)))$
24. $(\forall x ((f(x) \vee g(x)) \supset \neg h(x)) \wedge \forall x_1 ((g(x_1) \supset \neg i(x_1)) \supset (f(x_1) \wedge h(x_1)))) \supset \forall x_2 (i(x_2))$
25. $(\neg \exists x (f(x) \wedge (g(x) \vee h(x))) \wedge (\exists x_1 (i(x_1) \wedge f(x_1)) \wedge \forall x_2 (\neg h(x_2) \supset j(x_2)))) \supset \exists x_3 (i(x_3) \wedge j(x_3))$
26. $(\forall x ((f(x) \wedge (g(x) \vee h(x))) \supset i(x)) \wedge (\forall x_1 ((i(x_1) \wedge h(x_1)) \supset j(x_1)) \wedge \forall x_2 (k(x_2) \supset h(x_2)))) \supset \forall x_3 ((f(x_3) \wedge k(x_3)) \supset j(x_3))$
27. $\neg \exists y \forall x (f(x, y) \leftrightarrow \neg \exists z (f(x, z) \wedge f(z, x)))$

Fig. 4. Example Formulae

Example	Universe	Result	Swiss Time	Scottish Time
1.		Provable	14	18
2.		Provable	1388	1701
3.		Unprovable	15	21
4.		Provable	0.2	0.2
5.		Provable	0.1	0.1
6.		Provable	0.6	0.8
7.		Provable	11	14
8.		Provable	0.5	0.5
9.		Provable	4.3	4.3
10.		Unprovable	0.4	0.5
11.		Unprovable	24	10
12.		Provable	0.7	1.0
13.		Unprovable	4.5	3.2
14.		Provable	3.5	2.7
15.		Provable	50	57
16.	3	Provable	803	961
17.	2	Provable	7497	8450
18.	4	Provable	63	8.5
18.	5	Provable	146	15
19.	2	Provable	7.8	8.1
19.	3	Provable	18420	27
20.	2	Provable	1.1	2.1
20.	4	Provable	5.3	6.6
21.	2	Unprovable	8.6	10
21.	3	Unprovable	27	33
22.	2	Provable	366	22
22.	3	Provable	12320	514
23.	2	Provable	35	45
23.	3	Provable	2186	1407
24.	2	Unprovable	49	31
25.	2	Provable	10790	20
25.	4	Provable	NR	365
26.	2	Provable	3.4	5.8
26.	5	Provable	17	30
27.	2	Provable	10082	47

Table 1. Results and Timings (averages in milliseconds)

5 Conclusion

The use of a pared down history makes for a seemingly efficient means of loop detection for a theorem prover. However, as other intuitionistic theorem provers are written in different languages, are run on different machines and (in most cases) deal with first-order formulae, comparison is hard. An (incomplete) list of other intuitionistic theorem provers is: [2], [4], [8], [11], [12], [13], [14]. Of the two calculi given here, the one with the smallest history and the least checking (the Swiss one) can become inefficient (see example 27.) when delay in loop checking allows many extra branches to be pursued. In the Scottish style calculus the inefficiency of the increased history is more than counterbalanced by the early loop detection.

We have illustrated the use of the two history mechanisms on a particular calculus for intuitionistic propositional logic - one in which we are particularly interested, rather than because it is the best illustration. We anticipate similar advantages of the Scottish history mechanism in treatment of modal logics.

A final issue to be addressed is that of proof search. For this neither calculus is really suited, as they only find loop free proofs (plus a few more in the Swiss case). For details see [10].

References

1. Dyckhoff, R.: Contraction-free Sequent Calculi for Intuitionistic Logic. Journal of Symbolic Logic **57(3)** (1992) 795–807
2. Dyckhoff, R.: MacLogic implementation. Available from URL http://www-theory.dcs.st-and.ac.uk/∼rd/logic/soft.html
3. Dyckhoff, R., Pinto, L.: A Permutation-free Sequent Calculus for Intuitionistic Logic. University of St Andrews Research Report CS/96/9 (1996)
4. Dyckhoff, R., Pinto, L.: Implementation of a Loop-free Method for Construction of Counter-models for Intuitionistic Propositional Logic. University of St Andrews Research Report CS/96/8 (1996)
5. Gabbay, D.: Algorithmic Proof With Diminishing Resources, Part 1. Proceedings of the 1990 workshop Computer Science Logic, eds. Börger, E., Kleine Büning, H., Richter, M. M., Schönfeld, W.; Springer LNCS **533** (1991) 156–173
6. Girard, J.-Y.: A New Constructive Logic: Classical Logic. Mathematical Structures in Computer Science **1** (1991) 255–296
7. Herbelin, H.: A λ-calculus Structure Isomorphic to Gentzen-style Sequent Calculus Structure. Proceeding of the 1994 workshop Computer Science Logic, eds. Pacholski, L., Tiuryn, J.; Springer LNCS **933** (1995) 61–75
8. Heuerding, A., Jäger, G., Schwendimann, S., Seyfried, M.: Propositional Logics on the Computer. Proceedings of the 1995 international workshop on Theorem Proving with Analytic Tableaux and Related Methods (TABLEAUX '95), eds. Baumgartner, P., Hähnle, R., Posegga, J.; Springer LNAI **918** (1995) 310-323
9. Heuerding, A., Seyfried, M., Zimmermann, H.: Efficient Loop-Check for Backward Proof Search in Some Non-classical Propositional Logics. Proceedings of the 1996 international workshop on Theorem Proving with Analytic Tableaux and Related Methods (TABLEAUX '96), eds. Miglioli, P., Moscato, U., Mundici, D., Ornaghi, M.; Springer LNAI **1071** (1996) 210–225
10. Howe, J.M.: Theorem Proving and Partial Proof Search for Intuitionistic Propositional Logic Using a Permutation-free Calculus with Loop Checking. University of St Andrews Research Report CS/96/12 http://www-theory.cs.st-and.ac.uk/∼jacob/papers/tpil.html (1996)
11. Sahlin, D., Franzén, T., Haridi, S.: An Intuitionistic Predicate Logic Theorem Prover. Journal of Logic and Computation **2(5)** (1992) 619–656

12. Shankar, N.: Proof Search in the Intuitionistic Sequent Calculus. Proceedings of the 1992 international conference on Automated Deduction (CADE-13), ed., Kupar, D.; Springer LNAI **607** (1992) 522–536
13. Stoughton, A.: porgi:a Proof-Or-Refutation Generator for Intuitionistic propositional logic. http://www.cis.ksu.edu/∼allen/home.html
14. Tammet, T.: A Resolution Theorem Prover for Intuitionistic Logic. Available from the URL http://www.cs.chalmers.se/tammet/ (1996). This is a longer version of the paper in Proceedings of the 1996 international conference on Automated Deduction (CADE-13), eds. McRobbie, M. A., Slaney, J. K.; Springer LNAI **1104** (1996) 2-16
15. SICStus Prolog User's Manual. Swedish Institute of Computer Science (1993)

Acknowledgements

The author is indebted to Roy Dyckhoff for many useful discussions. The comments of the anonymous referees have also been very helpful and were greatly appreciated.