AN AUTOMATED SEARCH OF LINEAR INFERENCE RULES

Alvin Šipraga

December 13, 2012

Abstract

A linear inference is a logical inference where every variable occurs exactly once in the premiss and conclusion. In the paradigm of deep inference there are two well-studied linear inference rules, *switch* and *medial*. It has been shown that these are insufficient to generate all linear inferences under composition and *deep inference*. We consider the problem of finding the minimal inference not generated by *switch* and *medial* using computational methods and develop a tool to search for it.

1 Introduction

Deep inference is a methodology in proof theory in which rules are allowed to operate arbitrarily deep inside a formula (see [BT01]). This can be characterised as follows: Suppose $F(a \wedge b)$ is a monotone formula. We know $a \wedge b \to a \vee b$ is sound, and deep inference tells us also that $F(a \wedge b) \to F(a \vee b)$. This extends more generally to any sound inference $A \to B$ implying $F(A) \to F(B)$ is sound, and is stated and proved in §2 (Theorem 1).

There are three linear inference rules that are of core importance in deep inference: *mix*, *switch* and *medial* (studied in detail in [Str07]). Mix and switch have historical importance in the Gentzen formalism, while medial has been introduced by deep inference in order to gain locality in proof systems. They are, respectively[1]:

$$\mathsf{mix}\ \frac{a \wedge b}{a \vee b} \qquad \mathsf{s}\ \frac{a \wedge [b \vee c]}{(a \wedge b) \vee c} \qquad \mathsf{m}\ \frac{(a \wedge b) \vee (c \wedge d)}{[a \vee c] \wedge [b \vee d]}$$

Linear inferences such as these can be composed to prove soundness of other linear inferences. We can apply them to any part of a formula and form a derivation (the proof). For example, consider the linear inference $a \wedge [b \vee (c \wedge d)] \to [a \vee c] \wedge [b \vee d]$. We can prove soundness by composition of two of the above rules, switch and medial:

[1] The notation used here is common for deep inference, and is read top-down with each step expressing an application of a certain rule – so, any line "implies" any line below it.

$$\mathsf{s}\,\frac{\mathsf{m}\,\dfrac{a \wedge [b \vee (c \wedge d)]}{(a \wedge b) \vee (c \wedge d)}}{[a \vee c] \wedge [b \vee d]}$$

Each rule is necessary in some way to form such derivations. For example, working with just mix and medial (a "basis" $\{\mathsf{mix},\mathsf{m}\}$) and the linear inference

$$a \wedge [(b \wedge c) \vee (d \wedge e)] \to (a \wedge b \wedge c) \vee d \vee e, \tag{1}$$

consider (without loss of generality) all possible compositions of $\{\mathsf{mix},\mathsf{m}\}$ on the premiss (left hand side) of the inference (the "exhaustion" of $\{\mathsf{mix},\mathsf{m}\}$ on $a \wedge [(b \wedge c) \vee (d \wedge e)]$) up to permutations of the rules (cf. §3.2):

$$\mathsf{mix}\,\frac{\mathsf{mix}\,\dfrac{\mathsf{mix}\,\dfrac{a \wedge [(b \wedge c) \vee (d \wedge e)]}{a \vee (b \wedge c) \vee (d \wedge e))}}{a \vee b \vee c \vee (d \wedge e)}}{a \vee b \vee c \vee d \vee e} \quad \xrightarrow[\text{(see also §2.1)}]{\text{equivalent to}} \quad \mathsf{mix3}\,\frac{a \wedge [(b \wedge c) \vee (d \wedge e)]}{a \vee b \vee c \vee d \vee e}$$

$$\mathsf{mix}\,\frac{\mathsf{m}\,\dfrac{\mathsf{mix}\,\dfrac{a \wedge [(b \wedge c) \vee (d \wedge e)]}{a \vee (b \wedge c) \vee (d \wedge e))}}{a \vee ([b \vee d] \wedge [c \vee e])}}{a \vee b \vee d \vee c \vee e} \qquad \mathsf{mix2}\,\frac{\mathsf{m}\,\dfrac{a \wedge [(b \wedge c) \vee (d \wedge e)]}{a \wedge [b \vee d] \wedge [c \vee e]}}{a \vee b \vee d \vee c \vee e}$$

Notice that the conclusion (right hand side) of (1) appears nowhere in these derivations, so by exhaustion we can say that the inference is "unprovable" under the basis $\{\mathsf{smix},\mathsf{m}\}$. In fact, the inference can be proved by introduction of the switch rule:

$$\mathsf{mix}\,\frac{\mathsf{s}\,\dfrac{a \wedge [(b \wedge c) \vee (d \wedge e)]}{(a \wedge b \wedge c) \vee (d \wedge e)}}{(a \wedge b \wedge c) \vee d \vee e}$$

Straßburger shows in [Str12] that mix, switch and medial are not enough to prove *all* linear inferences, offering an example of a 36-variable such linear inference based on a propositional encoding of the pigeon hole principle – an extreme example. What is the smallest such linear inference, which can not be proved using only switch and medial? This is the motivating question of the project, and it is approached computationally. This fits into the broader project of understanding the space of linear inferences as a whole.

N.B. Later we exclude the mix rule given that we consider only *nontrivial* linear inferences. This is discussed in further detail later when the approach is presented (§3).

2 Preliminaries

Next we present definitions and results necessary to formalise the problem.

Definition 1. A *monotone formula* is a certain encoding of a boolean function, and is generated by the grammar

$$A ::= a \mid (A \wedge A) \mid [A \vee A]$$

where a is an *atom*. The *size* of a formula is the number of atom occurrences in it.

A formula A which has n distinct atoms will correspond to a boolean function $f_A : \{0,1\}^n \to \{0,1\}$ and computes it in the natural way, so we write $A(\sigma)$ to denote the output of $f_A(\sigma)$ for any assignment $\sigma \in \{0,1\}^n$.

Definition 2. A *read-once formula* (or *balanced formula*) is a formula in which each distinct atom occurs only once. That is, the size is equal to the number of distinct atoms in the formula. For example, the formulae

$$[a \vee b] \wedge c \wedge [d \vee e \vee f],\ a \wedge [e \vee f], \text{ and } a \wedge b \wedge d \wedge f$$

are all read-once formulae, whereas

$$(a \wedge b) \vee (a \wedge c),\ a \wedge b \wedge c \wedge [a \vee d], \text{ and } (a \wedge b \wedge c) \vee c \vee d \vee e$$

are not.

Definition 3. A *linear inference* is an expression $A \to B$ where A and B are monotone read-once formulae. We say that a linear inference $A \to B$ is *sound* if $A(\sigma) \leq B(\sigma)$ for all assignments $\sigma \in \{0,1\}^n$ (for A, B formulae of size n), and denote this by writing $A \Rightarrow B$ ("$A \to B$ is sound").

Definition 4. Define the equivalence relation $=$ by taking the reflexive, symmetric and transitive closure of the following equations:

$$\begin{aligned} &A \star B = B \star A, && \text{commutativity} \\ &(A \star B) \star C = A \star (B \star C), && \text{associativity} \\ &\text{if } A = B \text{ then } \xi\{A\} = \xi\{B\}, && \text{context closure} \end{aligned}$$

for $\star \in \{\wedge, \vee\}$ where A, B, C are formulae and $\xi\{\ \}$ any monotone context.

$=$ is itself a linear inference rule, and this definition is by no means trivial. We see in §3.2 that without this equivalence relation, an automated process of linear inference discovery may never return.

Definition 5. A *context* is a formula with one hole appearing in place of a subformula, e.g. $a \wedge \{\ \}$, $b \vee (a \wedge \{\ \}) \vee e$, and is denoted by $\xi\{\ \}$, $\zeta\{\ \}$, and so on. The hole can be filled with any formula, so for the first example $\xi\{c \vee d\}$ would correspond to $a \wedge [c \vee d]$.

Theorem 1 (Soundness of deep inference)**.** *Suppose that $A \to B$ is a sound linear inference, and that $\xi\{\ \}$ is a monotone context. Then $\xi\{A\} \to \xi\{B\}$ is sound.*

Proof. Observe that as $\xi\{\ \}$ is negation free, it can be written

$$\xi\{\ \} = P \star \xi'\{\ \}$$

for a particular (monotone) context $\xi'\{\ \}$ and formula P, where $\star \in \{\wedge, \vee\}$.

The context can be fully expanded in this way:

$$\begin{aligned}\xi\{\ \} &= P \star \xi'\{\ \} \\ &= P \star (Q \star \xi''\{\ \}) \\ &= P \star (Q \star (\cdots \star \zeta\{\ \})) \\ &= P \star (Q \star (\cdots \star (R \star \{\ \})))\end{aligned}$$

The last context $\zeta\{\ \}$ is a special case, and so we can proceed with structural induction on these contexts.

BASE CASE Let $\zeta\{\ \} := R \star \{\ \}$, where R is an arbitrary formula. We have the following two possible cases (note $A \Rightarrow B$):

$$\zeta\{\ \} = R \wedge \{\ \}, \text{ hence } \zeta\{A\} = R \wedge A \Rightarrow R \wedge B = \zeta\{B\},$$

$$\zeta\{\ \} = R \vee \{\ \}, \text{ hence } \zeta\{A\} = R \vee A \Rightarrow R \vee B = \zeta\{B\};$$

by definition of conjunction and disjunction. So $\zeta\{A\} \Rightarrow \zeta\{B\}$, and this is our base case.

INDUCTIVE HYPOTHESIS $\xi'\{A\} \Rightarrow \xi'\{B\}$ ($\xi'\{\ \}$ monotone).

INDUCTIVE STEP Let $\xi\{\ \} := P \star \xi'\{\ \}$, for some formula P. Here again we have two cases (now note $\xi'\{A\} \Rightarrow \xi'\{B\}$ by the inductive hypothesis):

$$\xi\{\ \} = P \wedge \xi'\{\ \}, \text{ hence } \xi\{A\} = P \wedge \xi'\{A\} \Rightarrow P \wedge \xi'\{B\} = \xi\{B\},$$

$$\xi\{\ \} = P \vee \xi'\{\ \}, \text{ hence } \xi\{A\} = P \vee \xi'\{A\} \Rightarrow P \vee \xi'\{B\} = \xi\{B\}.$$

Hence $\xi\{A\} \Rightarrow \xi\{B\}$, and the proof is complete. □

The condition that $\xi\{\ \}$ is monotone is important: consider a context $\xi\{\ \} := \neg\{\ \}$. Then simply consider the mix rule and apply de Morgan's laws: $\xi\{a \wedge b\} = \neg\,(a \wedge b) = \neg a \vee \neg b \not\Rightarrow \neg a \wedge \neg b = \neg\,[a \vee b] = \xi\{a \vee b\}$. Hence $\xi\{A\} \not\Rightarrow \xi\{B\}$ in this case.

3 Approach

There is no known way of finding the minimal linear inference unprovable by {mix,s,m} without progressively checking the provability (or lack thereof) under {mix,s,m} of every linear inference. It is known (and can be verified by hand) that {mix,s,m} is sufficient to prove essentially[2] all linear inferences of size 4 or less, so our search begins at linear inferences of size 5. This already presents many inferences to check for provability and the task becomes completely unfeasible to carry out by hand when considering linear inferences of greater size, so it was necessary to automate the process on a computer.

3.1 General automating principle

For a given (sound) linear inference $A \to B$, (un)provability is established by exhausting every possible composition of the rules {mix,s,m} on A and searching for some occurrence of B in these derivations. The inference $A \to B$ is provable if it occurs, and unprovable if it does not (so {mix,s,m} is insufficient to prove it, of course).

Since we seek the minimal such inference, it is necessary that when all linear inferences of size n are being checked for provability, we have already established that all linear inferences of size $n-1$ or less are provable, otherwise we can not possibly know if one of size n is the *minimal* case. So all linear inferences of size 5 were checked, then 6 and so on.

Finally, we ensure to only check sound linear inferences, as otherwise false positives would necessarily be returned.

3.2 Equivalence of formulae

The equivalence relation in Definition 4 is critical when programming a computer to check for a new linear inference. To illustrate this, suppose we don't consider everything modulo = and that, for example, the formulae $a \wedge b \wedge c \wedge d$ and $b \wedge a \wedge c \wedge d$ are distinct.

Clearly they are *invertible* (one implies the other, and vice versa), and so both of the following patterns are valid occurrences in some derivation performed in exhausting {s,m}:

$$\dfrac{\dfrac{\dfrac{\vdots}{a \wedge b \wedge c \wedge d}}{b \wedge a \wedge c \wedge d}\,\mathsf{comm.}}{\vdots} \qquad \dfrac{\dfrac{\dfrac{\vdots}{b \wedge a \wedge c \wedge d}}{a \wedge b \wedge c \wedge d}\,\mathsf{comm.}}{\vdots}$$

But – as the computer program doesn't know any better – these could be repeatedly chained together in a derivation and cause an infinite loop, meaning

[2] We will see why the mix rule is excluded, and how we characterise "relevant" inferences, in §3.3.

the program would never return. This example is for commutativity, but the principle extends more generally to the whole equivalence relation $=$ in Definition 4.

The following result tells us that this is actually sufficient and avoids *all* infinite loops of this nature:

Theorem 2. *Any sound linear inference rule that is invertible is a subrelation of* $=$.

Proof. See [Gur77] [Hei91]. □

This is sufficient because if a rule is non-invertible then it can not be repeatedly chained together and cause an infinite loop. In particular, since there are only finitely many boolean functions of a given size, our exhaustion must terminate.

3.3 A naïve algorithm and supermix

To begin with, the problem was not as specific and all linear inferences (as opposed to exclusively nontrivial ones) were considered under the whole basis $\{\mathsf{mix}, \mathsf{s}, \mathsf{m}\}$.

The original algorithm was an intuitive and straightforward means for testing all of the necessary linear inferences. It is outlined in pseudocode, where a pair `A,B` is analogous to $A \to B$:

```
function provable(A,B) {
    nextAs = exhaust(A);
    for each A' in nextAs {
        if(A'==B) return true;
        if(provable(A',B)) return true;
    }

    return false;
}

BFs = genbfs(n)
for A in BFs {
    for B in BFs {
        if(A!=B and sound(A,B)) {
            if(!provable(A,B)) {
                print "new inference A->B";
                exit;
            }
        }
    }
}
```

```
print "no new inferences under n variables";
exit;
```

1. generate the set $\mathcal{B}_n$ of all balanced formulae of size n (n variables)
2. go to next pair of formulae (A, B) where A and B are members of $\mathcal{B}_n$, do (3), or jump to (4) when complete
3. (a) if A is not equal to B, and $A \to B$ is sound, proceed, otherwise jump to (2)
   (b) generate the set $\mathcal{R}_A$ of resultant formulae of all possible single applications of the rules $\{\mathsf{mix}, \mathsf{s}, \mathsf{m}\}$ on the formula A.
   (c) if B is not a member of $\mathcal{R}_A$, and no pair (A', B) (where A' is a member of $\mathcal{R}_A$) is sound, then jump to (5), otherwise jump to (2)
4. return – *all linear inferences in* $\mathcal{B}_n$ *are provable*,
5. return – *the pair* (A, B) *is not provable*

On the face of it this algorithm will correctly determine whether or not $\{\mathsf{mix}, \mathsf{s}, \mathsf{m}\}$ is sufficient for size n linear inferences. A problem was encountered when it came to trivial inferences however.

Consider the inference $a \wedge [b \vee c] \to a \vee (b \wedge c)$. Exhausting all possible applications of $\{\mathsf{mix}, \mathsf{s}, \mathsf{m}\}$ (up to permutations of variables and rules),

$$\mathsf{mix}\,\frac{a \wedge [b \vee c]}{a \vee b \vee c} \qquad \mathsf{s}\,\frac{a \wedge [b \vee c]}{(a \wedge b) \vee c} \qquad \mathsf{mix}\,\frac{\mathsf{s}\,\dfrac{a \wedge [b \vee c]}{(a \wedge b) \vee c}}{a \vee b \vee c}$$

we see that the given inference is "new" by the algorithm's definition. In fact, this is an example of a linear inference rule known as *supermix*, defined in [Das12a]:

$$\mathsf{smix}\,\frac{a \wedge \bigvee_i^n b_i}{a \vee \bigwedge_i^n b_i}$$

where $n = 2$. The mix rule is also the simplest case of supermix, where $n = 1$:

$$\mathsf{mix}\,\frac{a \wedge b}{a \vee b}.$$

Intuitively inferences like $a \wedge [b \vee c] \to a \vee (b \wedge c)$ – while technically "new" – are not what we want, and the following definition neatly formalises this:

Definition 6. Let A, B be read-once formulae where $A \Rightarrow B$. $A \to B$ is trivial if, for some atom a, A and B are represented by the contexts $\xi\{a\}$ and $\zeta\{a\}$ respectively, and $\xi\{\top\} \Rightarrow \zeta\{\bot\}$.

Check for our example: let $\xi\{\ \} := a \wedge [\{\ \} \vee c]$ and $\zeta\{\ \} := a \vee (\{\ \} \wedge c)$ whereby $a \wedge [b \vee c] = \xi\{b\}$ and $a \vee (b \wedge c) = \zeta\{b\}$. $\zeta\{\top\} = a \wedge [\top \vee c] = a \wedge \top = a = a \vee \bot = a \vee (\bot \wedge c) = \zeta\{\bot\}$ so $\xi\{\top\} \Rightarrow \zeta\{\bot\}$ and hence triviality is established.

Theorem 3. *Suppose A, B are read-once formulae of size n, and that $A \to B$ is unprovably (by $\{\mathsf{s},\mathsf{m}\}$) sound, and trivial. Then there exist read-once formulae A', B' of size $n-1$ or less, such that $A' \to B'$ is unprovably sound, but nontrivial.*

Proof. See [Das12b]. □

Given this, it makes sense to *only* search for *nontrivial* linear inferences, as by Theorem 3 any trivial but unprovably sound linear inference can not be the minimal such linear inference. Moreover, it is not difficult to show that any trivial inference can be derived from nontrivial ones of smaller size when units are present in the language (see [Das12b]).

We now present a justification for excluding the mix and supermix rules from our set of existing linear inference rules.

Proposition 4. *The* mix *rule $a \wedge b \to a \vee b$ is trivial.*

Proof. Consider wlog the corresponding contexts $\xi\{\ \} := a \wedge \{\ \}$ and $\zeta\{\ \} := a \vee \{\ \}$.

$$= \frac{\xi\{\top\}}{a \wedge \top}$$
$$= \frac{}{a}$$
$$= \frac{a}{a \vee \bot}$$
$$= \frac{}{\zeta\{\bot\}}$$

□

Proposition 5. *The* supermix *rule $a \wedge \bigvee_i^n b_i \to a \vee \bigwedge_i^n b_i$ is trivial.*

Proof. Every b_i can be substituted for $\top$ in the premiss and $\bot$ in the conclusion, and then both formulae are logically equivalent to a. □

Theorem 6. *Suppose A, B, C, D are read-once formulae where $A \Rightarrow B \Rightarrow C \Rightarrow D$. Suppose $B \to C$ is trivial. Then $A \to D$ is trivial.*

Proof. Let A, B, C, D be represented by the contexts $\xi\{a\}, \xi'\{b\}, \zeta'\{c\}, \zeta\{d\}$ respectively (where a, b, c, d are trivialising atoms). $B \to C$ is trivial, so $\xi'\{\top\} \Rightarrow \zeta'\{\bot\}$. By soundness of $A \to B$ and $C \to D$, we have $\xi\{X\} \Rightarrow \xi'\{X\}$ and $\zeta\{X\} \Rightarrow \zeta'\{X\}$ for all formulae X. Then $\xi\{\top\} \Rightarrow \xi'\{\top\} \Rightarrow \zeta'\{\bot\} \Rightarrow \zeta\{\bot\}$. Hence $A \to D$ is trivial. □

Now not only are we not interested in trivial linear inferences, but Theorem 6 tells us that in fact any linear inference whose proof includes a mix (cf. Proposition 4) operation will itself be trivial. Thus we amend the algorithm to instead *only* check *nontrivial* linear inferences, and furthermore we exclude $\{\mathsf{mix}, \mathsf{smix}\}$ from our set of rules to exhaust, so we only check provability under $\{\mathsf{s}, \mathsf{m}\}$.

Triviality and the supermix rule are discussed in greater detail in [Das12b].

3.4 Limiting exhaustion of $\{\mathsf{s}, \mathsf{m}\}$

Building on the algorithm, we observe that there is redundancy in checking the complete exhaustion of the A in a pair (A, B). Let $\mathcal{R}_A$ denote the set of resultant formulae of all possible compositions of the rules $\{\mathsf{s}, \mathsf{m}\}$ on the formula A, and $\mathcal{B}_n$ denote the set of all read-once formulae of size n. At the moment we recursively check, for every A in $\mathcal{B}_n$, the provability of all (sound) linear inferences $A' \to B$ *for all* A' in $\mathcal{R}_A$. This is not needed, as we shall now show.

Definition 7. As introduced by [Str07], we write $A \xrightarrow{\{\mathsf{s},\mathsf{m}\}} B$ to denote a single application of $\{\mathsf{s},\mathsf{m}\}$ on A leading to B. Moreover, $\xrightarrow[\{\mathsf{s},\mathsf{m}\}]{*}$ denotes the transitive closure of $\{\mathsf{s},\mathsf{m}\}$, so $A \xrightarrow[\{\mathsf{s},\mathsf{m}\}]{*} B$ means "under some composition of $\{\mathsf{s},\mathsf{m}\}$ on A we get B". It is equivalent to saying that $A \to B$ is provable under $\{\mathsf{s},\mathsf{m}\}$.

Proposition 7. *Suppose some (sound) linear inference $B \to C$ is provable*[3]*. If $A \to B$ is provable, then $A \to C$ is also provable. If $C \to A$ is provable, then $B \to A$ is also provable.*

Proof. This is just chaining together two derivations. We have $A \xrightarrow[\{\mathsf{s},\mathsf{m}\}]{*} B$ and $B \xrightarrow[\{\mathsf{s},\mathsf{m}\}]{*} C$, hence:

$$A \xrightarrow[\{\mathsf{s},\mathsf{m}\}]{*} B \xrightarrow[\{\mathsf{s},\mathsf{m}\}]{*} C$$

and we have $A \xrightarrow[\{\mathsf{s},\mathsf{m}\}]{*} C$, so $A \to C$ is provable under $\{\mathsf{s},\mathsf{m}\}$. The latter claim is true by a symmetrical argument. □

So now consider the particular case where $A \xrightarrow{\{\mathsf{s},\mathsf{m}\}} B$ (one single application of $\{\mathsf{s},\mathsf{m}\}$). Then any inference $A \to C$ where $A \xrightarrow{\{\mathsf{s},\mathsf{m}\}} B \to C$ is provable if and only if $B \to C$ is provable. B and C are necessarily in $\mathcal{B}_n$ (as is A) in the context of the algorithm, and so the linear inference $B \to C$ will be checked for provability itself by the algorithm and the problem is reduced.

This means that for any unprovable $A \to B$ (A, B in $\mathcal{B}_n$), there is a C in $\mathcal{B}_n$ (which may in fact be A or B) such that $A \xrightarrow[\{\mathsf{s},\mathsf{m}\}]{*} C \to B$ where $C \to B$ can not be reduced any further in this fashion by $\{\mathsf{s},\mathsf{m}\}$. Then – while $A \to B$ may be a minimal (in terms of *size of formula and inference*) unprovable linear inference candidate – it is a more complicated version of a simpler linear inference $C \to B$, and this is the one seek.

So the algorithm is adjusted further. For any pair $A \Rightarrow B$ where A and B are members of $\mathcal{B}_n$: if $C \Rightarrow B$ for a C in $\mathcal{R}_A$, then we continue to the next pair (A, B), otherwise it must be that $A \to B$ can not be reduced any further by $\{\mathsf{s},\mathsf{m}\}$ and this is our new minimal linear inference. Note this means that if $\mathcal{R}_A$ is empty, then $A \to B$ is the new minimal linear inference, because A can not be reduced at all by $\{\mathsf{s},\mathsf{m}\}$.

[3] When we say provable, we now always mean provable under $\{\mathsf{s},\mathsf{m}\}$.

3.5 Loss of generality of read-once formulae

The naïve algorithm above will consider every pair of every possible formula permutations of read-once formulae, which has a lot of redundancy. In particular, consider the examples:

$$(a \wedge b) \vee c \to a \vee b \vee c,$$
$$(a \wedge c) \vee b \to a \vee b \vee c.$$

Syntactically these are different, however they represent the exact same linear inference and it doesn't make sense to check both for provability. The solution is as follows: for every pair of read-once formulae (A, B) (referring to $A \to B$), force A to be lexicographically ordered. Note that any inference $A \to B$ where A is not lexicographically ordered is simply a rearrangement of variables that corresponds to an existing inference where A is lexicographically ordered (so they are isomorphic). Thus the condition is sufficient for considering all possible linear inferences.

N.B. Forcing lexicographical ordering could equally be done on B and the above justification would hold.

4 Implementation

Given the computational enormity of the task, efficiency was essential to achieve results. A computer program (called Mimir) was developed with this in mind.

The programming language chosen was C, as it offers a good balance between efficiency and clarity. In particular, the ability to completely control memory allocation on the stack versus heap was particularly useful.

The primary development goal was efficiency, and so a lot of the resultant code for Mimir is useful in similar contexts. Given the short time of development though, there is certainly room for optimization. Fast applications of $\{\mathsf{mix}, \mathsf{s}, \mathsf{m}\}$ were implemented in conjunction with specialized formula-manipulation functions, and these should serve as an example if implementing other rewrite rules. Implementing a general language for rewrite rules (something similar to "regular expressions" with character strings) would likely not lend itself well to high performance computing however, unless a different (but applicable) storage of formulae was found. The current method of storage certainly works well for the existing hardcoded rules mix, switch and medial.

4.1 Internal storage of formulae

C provides only primitive datatypes and structures and so formula storage has to be standardized in some way internal to the program. Two methods were employed in the process: reverse Polish notation (RPN) storage, and, later (for reasons discussed below) bracketed Polish notation (BPN) storage.

4.1.1 Reverse Polish notation storage

Reverse Polish notation is simply a *postfix* $(a\ b\ \star)$ as opposed to *infix* $(a \star b)$ notation. The only operators we work with are $\wedge$ and $\vee$ and each take two arguments, so translation to RPN is straightforward.

$$a \vee b \vee (c \wedge d \wedge [e \vee f] \wedge g) \vee (h \wedge i) \rightsquigarrow a\ b\ c\ d\ e\ f \vee g \wedge \wedge \wedge h\ i \wedge \vee \vee \vee$$

The advantage to this is that a formula can be easily evaluated in linear time and there is no extra syntax necessary. Furthermore, it is very easy to generate all types of read-once formulae.

This is how Mimir was originally designed, but the equivalence relation described in Definition 4 complicated things. The liberal placing of operands (arguments) and operators in the notation led to a big overhead in testing the equivalence of two formulae. This reduced efficiency (the main argument for the storage method), and so it was scrapped.

4.1.2 Bracketed Polish notation storage

For this storage, operators took arbitrarily many operands and had limits defined by bracketing. Predictably, Polish notation refers to *prefix* notation $(\star\ a\ b)$. Taking the above example again:

$$a \vee b \vee (c \wedge d \wedge [e \vee f] \wedge g) \vee (h \wedge i) \rightsquigarrow \vee[a\ b\ \wedge [c\ d\ \vee [e\ f]\ g]\ \wedge [h\ i]]$$

This notation is easier to read, and it was written internally to ignore the order of operands in checking for equivalence. This was much easier because each operator's scope could be checked recursively, leading to an easy to understand algorithm much more efficient than any such checks for the former storage.

The "open bracket" [is syntactic sugar and internally it was never added, instead it being implicit with the occurrence of any operator.

4.2 Limits of computability

At $n \geq 7$ and beyond (n being the size of linear inferences), the computation took far too long on a regular desktop computer. Further details relating to the results are discussed in section 5, while this section deals with sourcing more computing power.

Access was granted to the University of Bath High Performance Computing (HPC) Facility, specifically an 800-core Intel Xeon E5430 2.66GHz machine at 2GB/core of 667MHz memory, with DDR Infiniband interconnect. Jobs utilizing up to 64 cores for up to 12 hours were permitted.

The operating procedure of Mimir was straightforward enough to split up into multiple tasks, and so this was done with support of the OpenMPI message passing library to support communication between cores. Because of the 12 hour limit, basic checkpointing was also implemented alongside this.

Having generated all read-once formulae, they are stored to file. This is the first instance of Mimir that is to be run, and is a relatively quick process. Then

a batch job script will call any number of subsequent processes which will read the given file and store in virtual memory all lexicographically ordered formulae (which will be the A in $A \to B$, and does not represent any sizable memory overhead even for a relatively high number of variables) and an equal fraction of all formulae in file, depending on the number of total processes (which will be the B in $A \to B$). Iteration is then done over these two sets just as in the algorithm described in section §3.3.

5 Results

Mimir was successfully run on all linear inferences of size 5 and 6, and showed that there are unprovable nontrivial linear inferences of such size. Running for 7 or more posed a greater difficulty that is detailed in section §4.2.

Anupam Das found an example of a nontrivial linear inference of size 10 which could not be proved using $\{\mathsf{s}, \mathsf{m}\}$, which follows:

$$\frac{[a \vee (b \wedge c)] \wedge [(d \wedge e) \vee (f \wedge g)] \wedge [(h \wedge i) \vee j]}{([d \vee h] \wedge [a \vee (e \wedge i)]) \vee ([(b \wedge f) \vee j] \wedge [c \vee g])}$$

A construction and explanation for how it was found is detailed in [Das12a].

Thus the task remained to run for sizes 7, 8 and 9 to verify if this was in fact the minimal linear inference sought.

Due to code bugs and lack of time, Mimir hasn't yet been run for sizes 7, 8 or 9, but in doing so the minimal linear inference will be found. A run for size 9 linear inferences may require further optimization to return a result in any realistic time frame, however.

6 Further research

These results warrant further research. In particular, the structure of new rewrite rules can themselves be studied, as is seen in [Str07]. It would be worth carrying out a similar study on the new rule given in §5.

There are also further open questions with regards to the overall approach of this project:

- Is the linear inference given in §5 the minimal nontrivial linear inference that is unprovable by $\{\mathsf{s},\mathsf{m}\}$?
- Is there an infinite number of nontrivial linear inferences that can not be proved by composition of linear inference rules of smaller size, i.e. is there not a finite basis of the fashion $\{\mathsf{s},\mathsf{m}\}$ under which linear inferences of any finite size can be proved?
- If there is an infinite number, is there some kind of systematic pattern to their structure? Can the nth such linear inference be readily derived by some algorithm that does not require brute-force?

7 Mimir source and availability

Full source, documentation and up-to-date development information can be fount at `http://arcturus.su/mimir/`.

References

[BT01] Kai Brünnler and Alwen Fernanto Tiu. A local system for classical logic. In R. Nieuwenhuis and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning (LPAR)*, volume 2250 of *Lecture Notes in Computer Science*, pages 347–361. Springer-Verlag, 2001. `http://www.iam.unibe.ch/ kai/Papers/lcl-lpar.pdf`.

[Das12a] Anupam Das. Linear inferences and derivations 1, 2012. `http://article.gmane.org/gmane.science.mathematics.frogs/612`.

[Das12b] Anupam Das. The size of linear derivations in deep inference, 2012. `http://www.anupamdas.com/items/LinCompDI/LinCompDI.pdf`.

[Gur77] V. A. Gurvich. On repetition-free boolean functions. *Uspekhi Mat. Nauk*, 32(1(193)):183–184, 1977.

[Hei91] Rafi Heiman. Randomized decision tree complexity for read-once boolean functions, 1991.

[Str07] Lutz Straßburger. A characterisation of medial as rewriting rule. In Franz Baader, editor, *Term Rewriting and Applications—18th International Conference, RTA*, volume 4533 of *Lecture Notes in Computer Science*, pages 344–358. Springer-Verlag, 2007. `http://www.lix.polytechnique.fr/ lutz/papers/CharMedial.pdf`.

[Str12] Lutz Straßburger. Extension without cut. *Annals of Pure and Applied Logic*, 163(12):1995–2007, 2012. `http://www.lix.polytechnique.fr/ lutz/papers/psppp.pdf`.