Automating search of linear inferences in propositional logic

Alvin Šipraga

University of Bath

13 December, 2012

Linear inferences

A *linear inference* is a sound implication $A \to B$, free of negation, and where every variable occurs exactly once in the premiss and conclusion (i.e. it is *balanced*).

Some examples:

- $(a \wedge b) \vee c \to a \vee b \vee c$
- $(a \wedge e \wedge f) \vee (b \wedge [d \vee c]) \to (a \wedge e \wedge f) \vee b \vee d \vee c$

are linear inferences. However:

- $[a \vee b] \wedge [a \vee c] \to a \vee b$ (unbalanced)
- $a \vee b \to a \wedge b$ (unsound)
- $\neg\,(a \wedge b) \vee c \to \neg a \vee \neg b \vee c$ (negation)

are not.

Linear inferences

Why negation free?

If an inference is balanced, the negation free condition we set is for clarity. Any instance of negation can be reduced by de Morgan's laws to atom-by-atom negation. Since each atom occurs only once, we can then replace them.

$$\dfrac{\dfrac{\neg(a \wedge [b \vee c])}{\neg a \vee \neg\,[b \vee c]}}{\neg a \vee (\neg b \wedge \neg c)}$$

Note that balancedness ensures *either* only x or $\neg x$ occur – not both. In the above example we would replace $\neg a$, $\neg b$, $\neg c$ with fresh variables a', b', c' respectively for an equivalent *negation free* formula.

Three linear inference rules

Within deep inference, there are three important linear inference rules, some of which may be familiar: *mix*, *switch* and *medial*.

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

- These are strict implications
- Composition of these rules is the basis of the study

Example

$a \wedge [b \vee (c \wedge d)] \to [a \vee c] \wedge [b \vee d]$

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

Three linear inference rules

Building a basis

We would like to understand the space of linear inferences better. One way of looking at this is to treat the three rules we introduced as a basis for generating linear inferences. But are these three rules sufficient to span all linear inferences?

Answer: no. Straßburger offered the following counterexample with 36 variables:

$$\bigwedge_{1\leq i\leq 18} [a_i \vee b_i] \rightarrow ([a_1 \vee a_2 \vee a_3] \wedge [a_7 \vee a_8 \vee a_9] \wedge [a_{13} \vee a_{14} \vee a_{15}])$$
$$\wedge([b_1 \vee a_4 \vee a_5] \wedge [b_7 \vee a_{10} \vee a_{11}] \wedge [b_{13} \vee a_{16} \vee a_{17}])$$
$$\wedge([b_2 \vee b_4 \vee a_6] \wedge [b_8 \vee b_{10} \vee a_{12}] \wedge [b_{14} \vee b_{16} \vee a_{18}])$$
$$\wedge([b_3 \vee b_5 \vee b_6] \wedge [b_9 \vee b_{11} \vee b_{12}] \wedge [b_{15} \vee b_{17} \vee b_{18}])$$

which can not be generated using the mix, switch and medial rules.

Three linear inference rules

Approach

This raises the question:

- What is the greatest number of variables in a linear inference where we can be sure mix, switch and medial provide a complete basis?
- What is the simplest linear inference for which switch, mix and medial are insufficient?

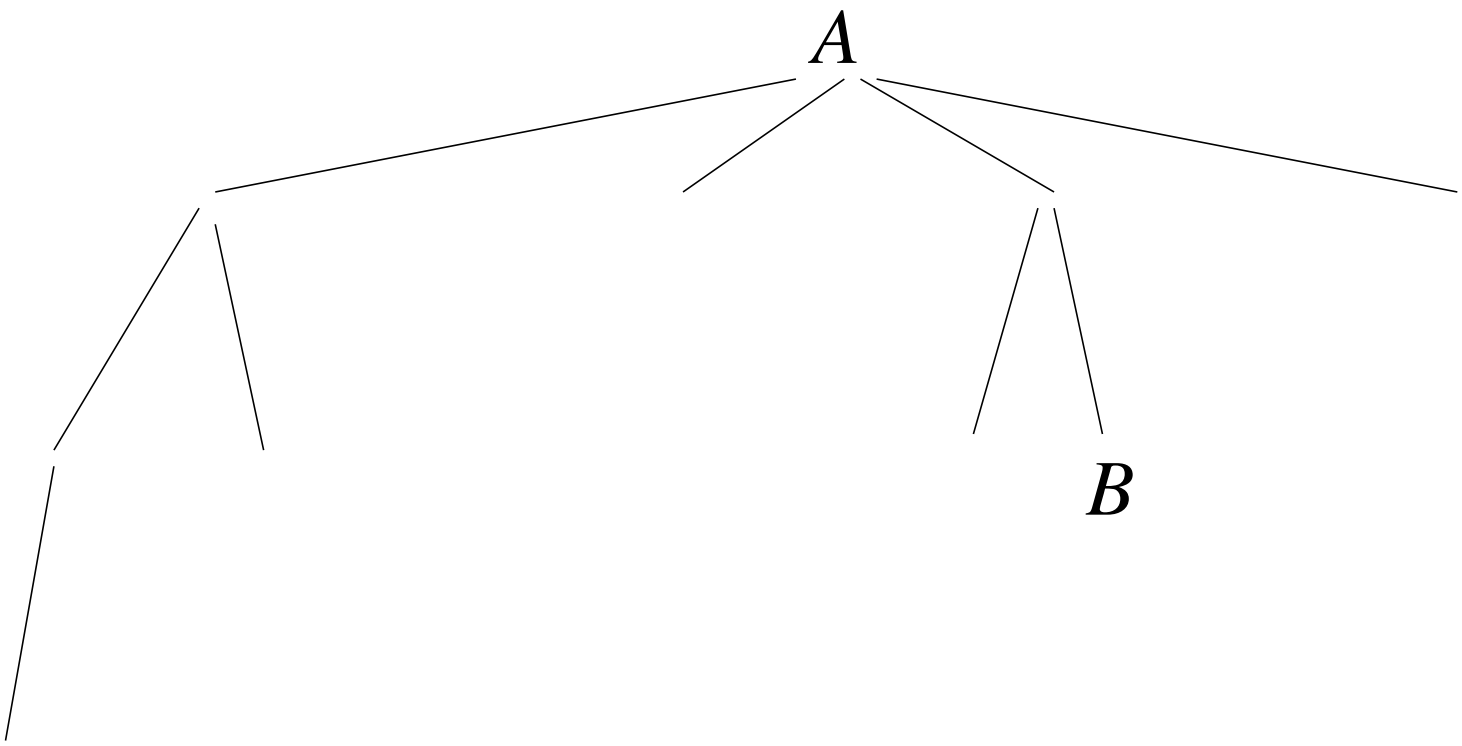
- No known theoretical method for answering this.
- But we can apply a brute force method by freely applying the three rules.

So we can think about it like this: for each linear inference $A \to B$, is it provable by mix, switch and medial? Supposing we do this systematically by considering first all formulae of size 3, then 4, and so on, we will answer the question.

Three linear inference rules

Example

$A \to B$



A naïve algorithm

```
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;
```

A naïve algorithm

There are a few considerations we should make, in particular whether the algorithm will always return. In fact it its current state it won't unless we clearly define what we mean for two formulae to be equal and how to implement it.

A naïve algorithm

Ensuring termination

Consider:

$$a \wedge b \wedge c \wedge d \qquad b \wedge a \wedge c \wedge d$$

Clearly equal up to commutativity.

A possible scenario could be something of this form:

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

Could something like this still happen in other cases?

A naïve algorithm

Ensuring termination

Definition

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

$$A \star B = B \star A, \qquad \text{commutativity}$$
$$(A \star B) \star C = A \star (B \star C), \qquad \text{associativity}$$

for $\star \in \{\wedge, \vee\}$ where A, B, C are formulae.

Theorem (Gurvich)

Any sound linear inference rule that is invertible is a subrelation of $=$, where $=$ is equivalence up to commutativity and associativity.

So consider formulae equal up to $=$, which ensures termination by the above theorem.

A naïve algorithm

Trivial linear inferences

Consider the inference $a \wedge [b \vee c] \to a \vee (b \wedge c)$. Exhausting all possible applications of mix, switch and medial (up to permutation), we have:

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

The conclusion $a \vee (b \wedge c)$ does not appear here, so as far as the algorithm is concerned, this is a new inference!

How do we characterise this unwanted type of inference?

$$\frac{a \wedge [\top \vee c]}{a \vee (\bot \wedge c)}$$

"Trivial at b."

A naïve algorithm

Trivial linear inferences

Theorem

If $\rho : F\{a\} \to G\{a\}$ is trivial at a, then:

1. *ρ can be reduced to a linear inference ρ' with strictly fewer variables;*
2. *ρ can be derived from ρ' using units.*

Example (mix)

$$\frac{a \wedge b}{a \vee b} \quad \rightsquigarrow \quad \frac{a \wedge \top}{a \vee \bot} \quad \rightsquigarrow \quad \text{mix is trivial (at } b\text{)}$$

Proposition

Suppose A, B, C, D are balanced formulae where $A \Rightarrow B \Rightarrow C \Rightarrow D$. Suppose $B \to C$ is trivial. Then $A \to D$ is trivial.

A naïve algorithm

```
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;
```

Optimisation

The algorithm presented is *slow*. Next are two improvements that reduce redundancy, which we can summarise as follows:

- Why check both of the following inferences?

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

- Does `provable()` need to be recursive? $A \rightarrow B$:

$$\{\mathsf{s,m}\}\ \frac{A}{A'}$$
$$*\ \frac{}{B}$$

$A' \rightarrow B$ would be checked anyway.

A naïve algorithm

Final version

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

    return false;
}

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

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

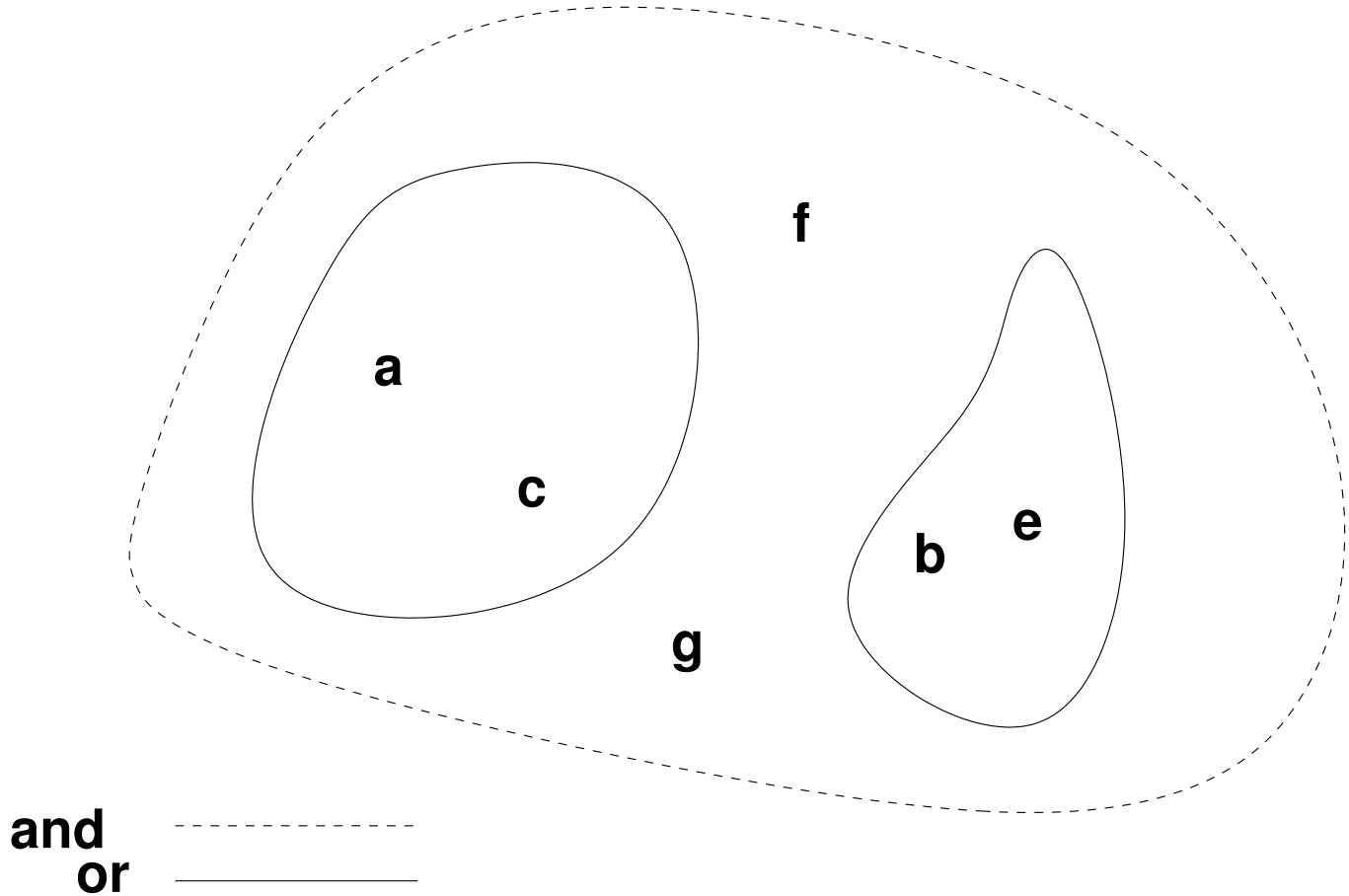
Implementation

- C was chosen to write the program to make the most of limited memory/computing power.
- Storage of formulae proved to be very important.
- The mix, switch and medial rules were implemented ad hoc - automatic implementation is a much harder problem.
- Execution became unrealistic on a regular desktop machine for checks on all linear inferences of sizes 7 or more.

Implementation

Storage of formulae

$$[a \vee c] \wedge [b \vee e] \wedge f \wedge g$$



Summary

- Generating all linear inferences of up to 3 variables by hand is simple.
- We have found that switch and medial also span all linear inferences of up to 6 variables.
- Simpler linear inference that can not be generated:

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

- Reduces the problem to searching for linear inferences of between 7 and 9 variables.
- It is known that proof search in a certain deep inference system (KS) can be efficiently reduced to the problem of $\{$s,m$\}$ derivability.
- The implementation includes a small library of functions for manipulating formulae stored in the specified format – could perhaps be a framework for a more general deep inference tool.

Thanks for listening!