Proof Support for Hybrid Systems Verification, III

Advanced Refinements to Resolution

Lawrence C Paulson, Computer Laboratory, University of Cambridge

A more advanced Knuth-Bendix ordering

- KBO is popular in resolution theorem proving
  - it is efficient
  - and prefers **smaller** terms
- but if we want to eliminate functions like $\mathbf{exp}$, we must be prepared to accept **larger** terms

Example: a lower bound for $\exp$

We have the simple $1 + x \leq \exp x$

from which we obtain, since $x < y$ denotes $\neg(y \leq x)$,

the clauses

$$\boxed{y \leq \exp x} \vee y > 1 + x$$

$$\boxed{\exp x > y} \vee 1 + x \leq y$$

the $\exp$ literals dominate, according to KBO, and are given priority. **BUT …**

Another lower bound for $\exp$

$$\left(1+\frac{x}{3}+\frac{x^2}{18}+\frac{x^3}{162}+\frac{x^4}{1944}+\frac{x^5}{29160}\right)^3 \leq \exp x$$

from which we obtain…

$$y \leq \exp x \vee y > \left(1+\frac{x}{3}+\frac{x^2}{18}+\frac{x^3}{162}+\frac{x^4}{1944}+\frac{x^5}{29160}\right)^3$$

KBO doesn't like the multiple occurrences of x!

Also, bounds can be arbitrarily "heavier" than a function call!

Possible extensions to KBO, I

Term weights could be *transfinite ordinals*:

$$
\begin{aligned}
1 < 2 < \cdots < n < \cdots &< \omega \\
< \omega + 1 < \cdots < \omega + n < \cdots &< \omega 2 \\
< \omega 2 + 1 < \omega 3 < \cdots < \omega n < \cdots &< \omega^2 \\
< \omega^2 + \omega < \omega^3 < \omega^4 < \cdots < \omega^n < \cdots &< \omega^\omega < \cdots
\end{aligned}
$$

Intuitively, these are *nested lists* of integers, giving us an unlimited system of priorities for function symbols.

But weight calculations would become expensive!

A compromise: weights could be **simpler** ordinals

E.g. *pairs* of natural numbers, equivalent to the ordinals below $\omega 2$

runtime would still be *doubled*

Possible extensions to KBO, II

Refining the concept of *number of occurrences*

what if we counted $\mathbf{exp}\, x$ as having 150 occurrences of x?

This is called a *subterm coefficient* for $\mathbf{exp}$, a simple modification of the weight calculation.

ω would be better, but ordinal calculations are costly.

Weights are determined by *experimentation*, coefficients simply by inspection

Next idea: splitting

Case-splitting in resolution

What is it?

splitting a clause in half —
doing two separate refutations

When can you do it?

only when variables
can be *partitioned*

Why *would* you do it?

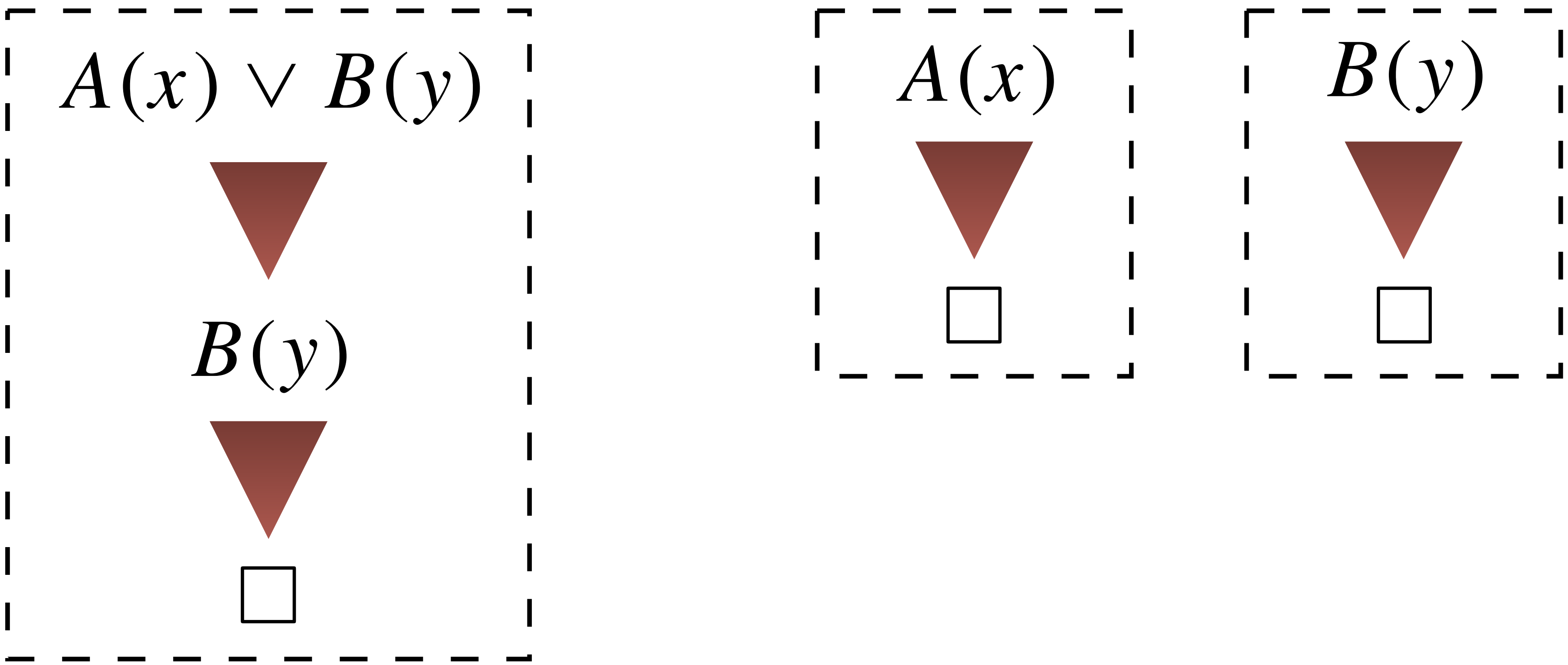
shorter clauses yield shorter proofs

Why in MetiTarski?

hand simulation suggested
it was necessary

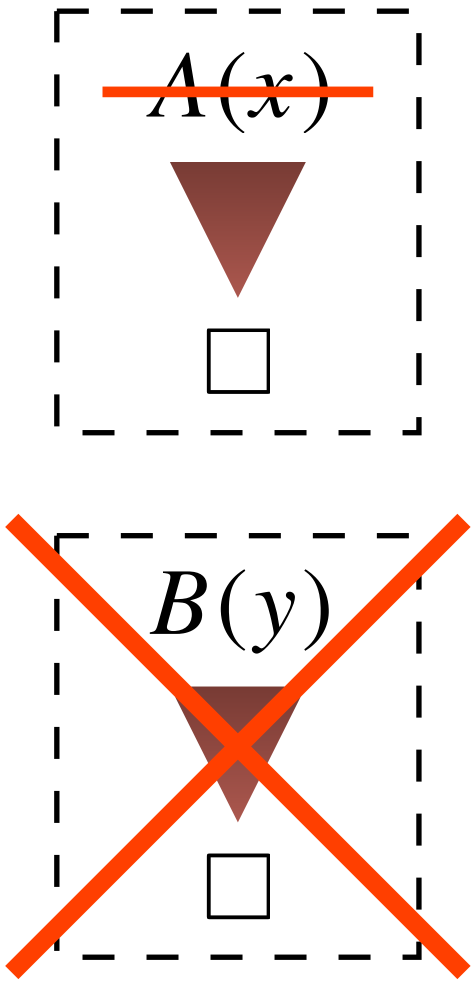
The clause $A(x) \vee B(y)$

is logically equivalent to $(\forall x.\, A(x)) \vee (\forall y.\, B(y))$



- Splitting isn't necessary but it's often powerful
- Both halves of the clause must be **nontrivial**
- Different from SAT-solving, but shared ideas

Intelligent backtracking



- The first case is proved without using $A(x)$
- Therefore the proof does not need $A(x) \lor B(y)$
- *So the second proof is skipped!*
- (a key refinement in SAT-solving, too)

How to implement splitting?

The **hard** way, with a “split stack” to manage backtracking and restore data structures

- first implemented in SPASS

  Fietzke A., Weidenbach C. Labelled splitting. In: Armando A., Baumgartner P., Dowek G. (eds) *Automated Reasoning: IJCAR* (2008), 459–474. Springer LNCS 5195.

The **easy** way, with new predicate symbols instead of backtracking

- No fancy data structures: resolution does all the work

  Riazanov, A., Voronkov A.: Splitting without backtracking. In: *International Joint Conference on Artificial intelligence (IJCAI-17)* — Volume 1, pp. 611– 617 (2001).

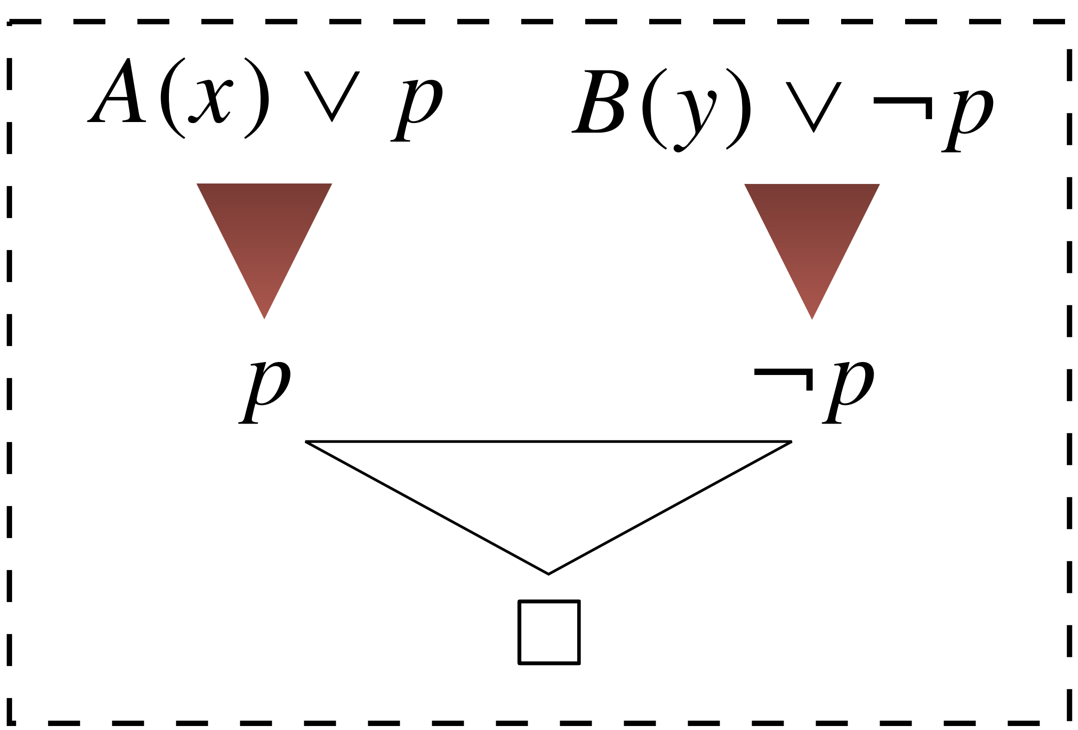
Splitting **without** backtracking

The clause $A(x) \vee B(y)$ is split into two clauses:

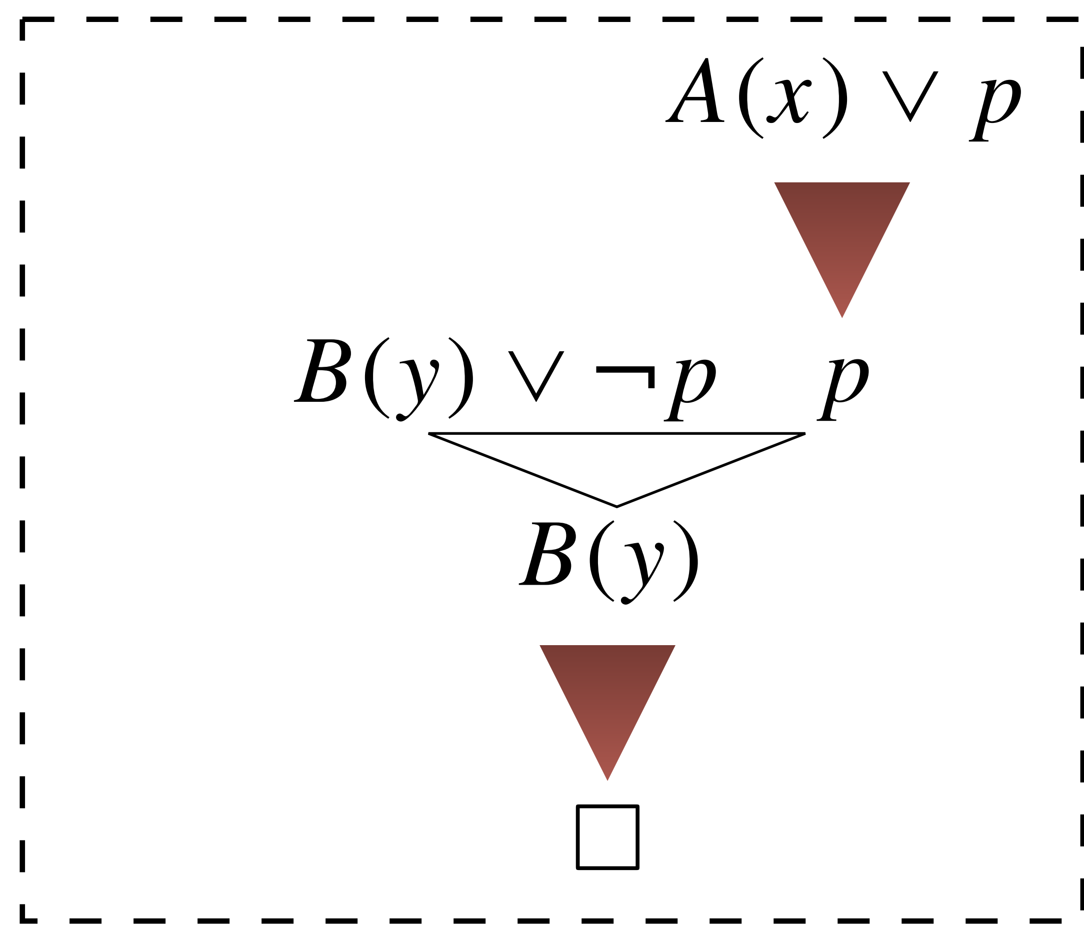
$$A(x) \vee p \qquad B(y) \vee \neg p$$

And then, give it to normal resolution… but will it work?

parallel derivation (bad!)



serial derivation



Ordering constraints for splitting

- The "split labels" p are made minimal so that they persist
- But negated labels $\neg p$ are made maximal
  - then a clause containing $\neg p$ can *only* resolve with p
  - **completely blocking** the clause until p is proved

Further refinements of splitting

- *Hyper-splitting*: splitting the clause into three or more parts in a single step
- *Naming heuristic*: reusing a label when splitting on the same formula
- MetiTarski-specific aspects:
    - only *ground* clauses are split (these are common)
    - both cases of the split must involve a *special function*

Next idea: introducing decision procedures

Combining resolution with a decidable theory

- An old issue: resolution is *good with quantifiers* but **bad** with arithmetic and other theories

- Many have tried to combine resolution with decision procedures. Two notable examples:

  Bonacina MP, Lynch CA, Moura L. On deciding satisfiability by theorem proving with speculative inferences. *JAR* 47, 2 (2011), 161–189.

  Prevosto V, Waldmann U. SPASS+T. In Sutcliffe G, Schmidt R; Schulz S. Eds, *ESCoR: Empirically Successful Computerized Reasoning* (2006), 18–33.

Literal deletion by decision procedure

- A list of “relevant” clauses is maintained. (For MetiTarski: containing just + − × = ≤ and no variables.)
- Every literal of each new clause is examined.
- A literal will be *deleted* if—according to the decision procedure—it is *inconsistent* with its context.
- MetiTarski also uses the decision procedure to detect *redundant* clauses (those deducible from known facts).

Perhaps other decision procedures could similarly be integrated with resolution

Examples of algebraic literal deletion

- *Unsatisfiable* literals such as $p^2 < 0$ are deleted.
- If $x(y+1) > 1$ has previously been deduced, then $x=0$ will be deleted.
- A literal's **context** includes the *negations of adjacent literals* in the same clause
  - $z > 5$ is deleted from $z^2 > 3 \lor z > 5$
  - … because $\exists z\ [z^2 \leq 3 \land z > 5]$ reduces to FALSE.

Next idea: low-level support for arithmetic expressions

Horner normal form: a canonical form for arithmetic

a *recursive* representation of polynomials

$$a_n x^n + \cdots + a_1 x + a_0$$
$$= a_0 + x(a_1 + x(a_2 + \cdots x(a_{n-1} + x a_n))$$

$$3xy^2 + 2x^2yz + zx + 3yz$$
$$= [y(z3)] + x([z1 + y(y3)] + x[y(z2)])$$

A polynomial in x

where the coefficients are polynomials in y …

Highly unnatural, but a basis for simplification

Formula simplification: finishing up

- Finally we simplify the Horner normal form, using laws like $0+z = z$ and $1\times z = z$.

- The maximal "foreign" term, say ln *E*, is isolated (if possible) on one side of an inequality.

- Nested quotients are flattened to *rational functions*:

$$\left(\frac{y}{x}\right)\frac{1}{\left(x+\frac{1}{x}\right)} = \frac{x^2}{y(x^2+1)}$$

One last thing: dividing out products

- Given a clause of the form $f(t) \cdot u \leq v \vee C$

- deduce the *three* clauses

$$f(t) \leq v/u \vee u \leq 0 \vee C$$

$$0 \leq v \vee u \neq 0 \vee C$$

$$f(t) \geq v/u \vee u \geq 0 \vee C$$

- Actually a new inference rule! But needed when a function is **multiplied** by another term.

Summary

Subterm coefficients are a useful and cheap extension of KBO.

Splitting can be done simply by hacking the ordering.

Literal deletion is one way to augment resolution with a decision procedure.

Simplification must be provide a *canonical form* for terms of the theory.