Using inference to improve search on SAT problems

Lyndon Drake

lyndon@cs.york.ac.uk

AI Group, Computer Science Department,
University of York, United Kingdom

Introduction

- SAT: propositional satisfiability problem
- Archetypal NP-complete problem (Cook, 1971)
- Numerous practical applications, including planning, quasigroup completion, and model checking
- My research: evaluating techniques for using inference to improve search

Outline

- Examples of SAT instances and an application
- Using backtracking search to solve SAT
- Adding inference to search
- Evaluating neighbour resolution
- Future work

Satisfiable SAT instance

$$\begin{array}{ccc} a & \vee & \neg b \\ b & \vee & c \\ \neg c & \vee & d \end{array}$$

Alternative form:

$$\Sigma = (a \vee \neg b) \wedge (b \vee c) \wedge (\neg c \vee d)$$

Satisfying assignment:

$$A = \{a \mapsto T,\ \ b \mapsto T,\ \ c \mapsto F\}$$

Unsatisfiable SAT instance

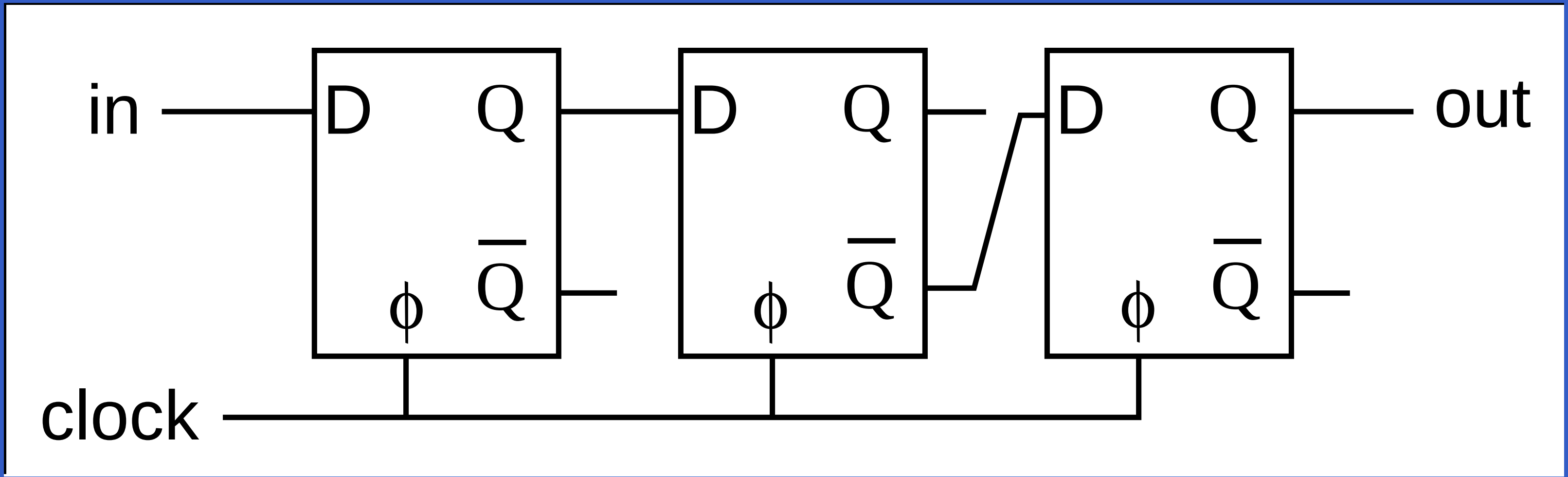
$$
\begin{array}{ccccc}
a & \vee & \neg b & \vee & \neg c \\
a & \vee & b & \vee & c \\
\neg a & \vee & \neg b & \vee & \neg c \\
\neg b & \vee & d & & \\
\neg b & \vee & \neg d & & \\
a & \vee & c & \vee & d \\
\neg b & \vee & c & \vee & \neg d
\end{array}
$$

$$
\begin{array}{ccccc}
a & \vee & b & \vee & d \\
a & \vee & \neg c & \vee & \neg d \\
\neg a & \vee & b & \vee & d \\
b & \vee & \neg c & \vee & \neg d \\
\neg a & \vee & c & \vee & d \\
b & \vee & c & \vee & \neg d
\end{array}
$$

BMC example: shift register

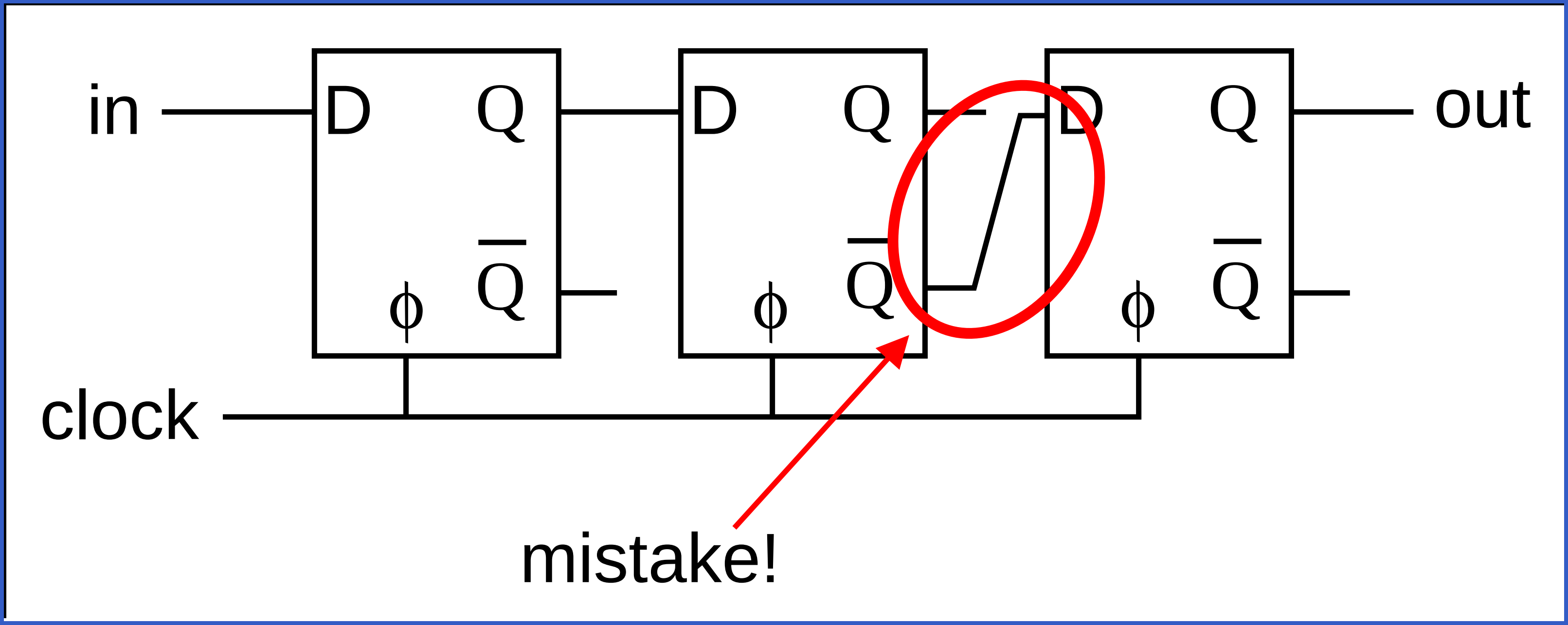
BMC example: shift register

- “I think this is a shift register” (The model)



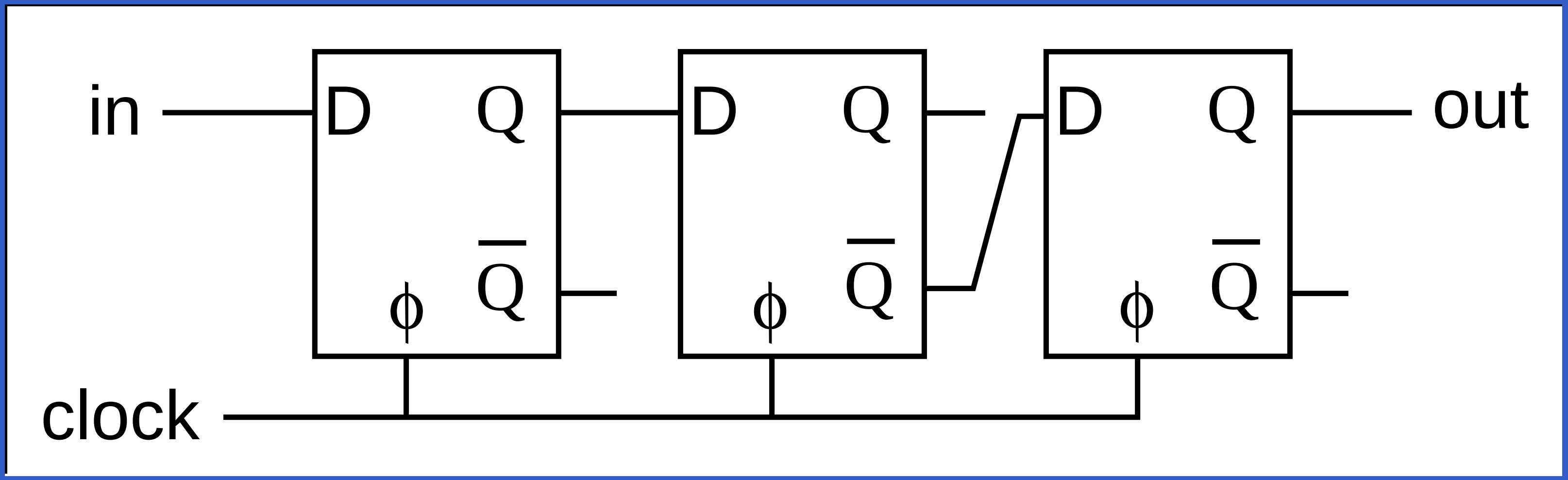
BMC example: shift register

- “I think this is a shift register” (The model)



BMC example: shift register

- “I think this is a shift register” (The model)



- “This is what a shift register does” (The specification):
  Data on the input propagates unchanged to the output

Bounded model checking

- Bounded model checking is used to test a model's conformance to a specification
- Used for testing:
  - safety critical systems
  - digital circuit designs
- State-of-the art for circuit model-checking is to solve via a SAT encoding

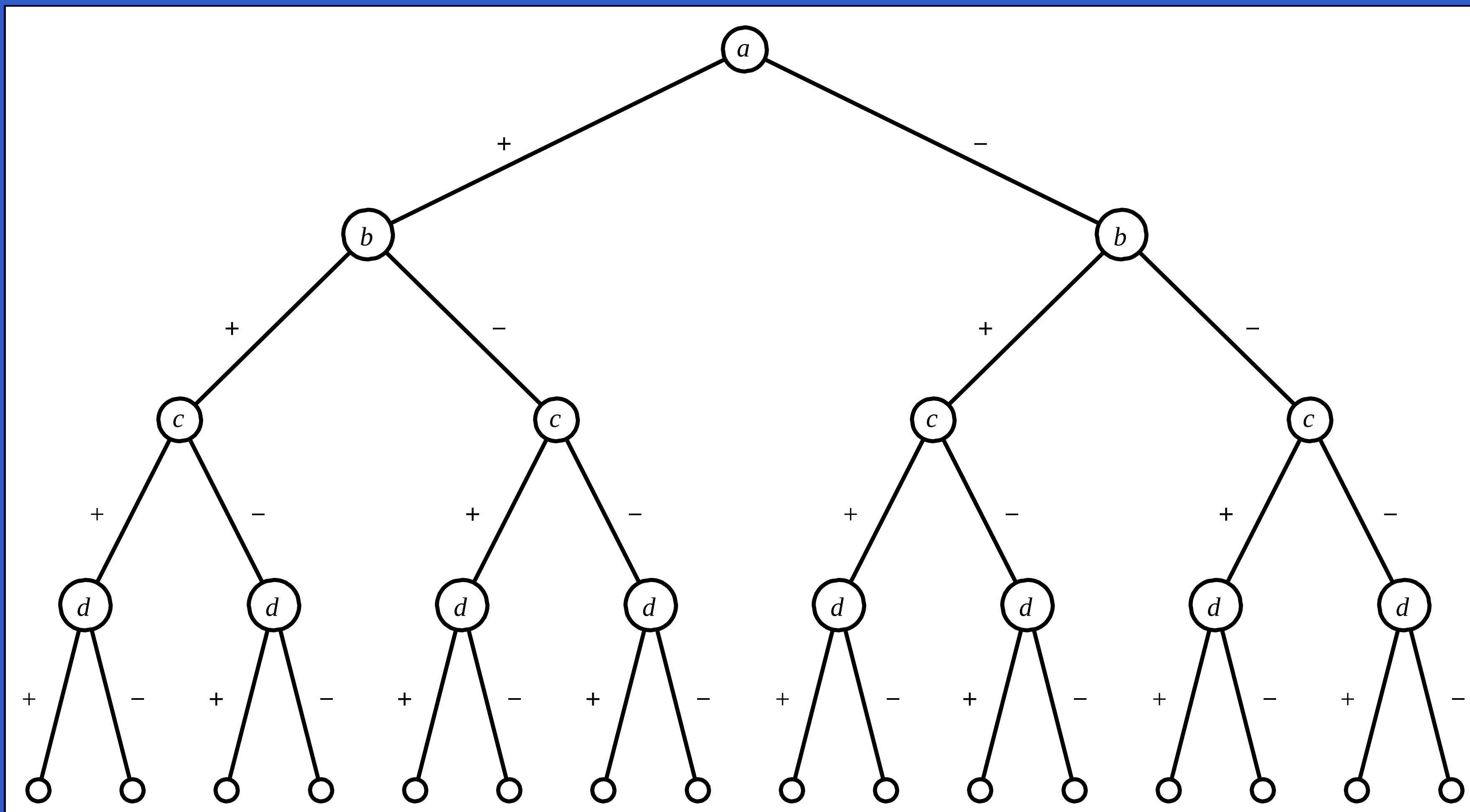
Encoding BMC as SAT

- Encode the specification in a temporal logic
- Map both the model and the specification to a SAT formula
- Give the formula to a SAT solver
- If a satisfying assignment exists, there is a bug
    - well, maybe. . .

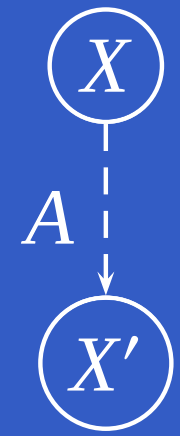
Solving SAT problems

- The Davis-Putnam method
  - Uses resolution to eliminate variables
- Resolution is intractable and impractical for many problems
  - E.g. *best* case exponential space complexity on pigeon-hole problems
- DLL (Davis-Logemann-Loveland)
  - Replaces resolution with branching
  - Complete backtracking search
  - Basis of the fastest available solvers

Search space for unsatisfiable instance



Assignment during search



- $X = (a \vee \neg b \vee \neg c)$
- $A = \{a \mapsto F,\ \ b \mapsto T\}$
- $X' = (\neg c)$

Example of unit propagation

Given:

- $X = (a \vee \neg b \vee \neg c)$
- $A = \{a \mapsto F,\ \ b \mapsto T\}$

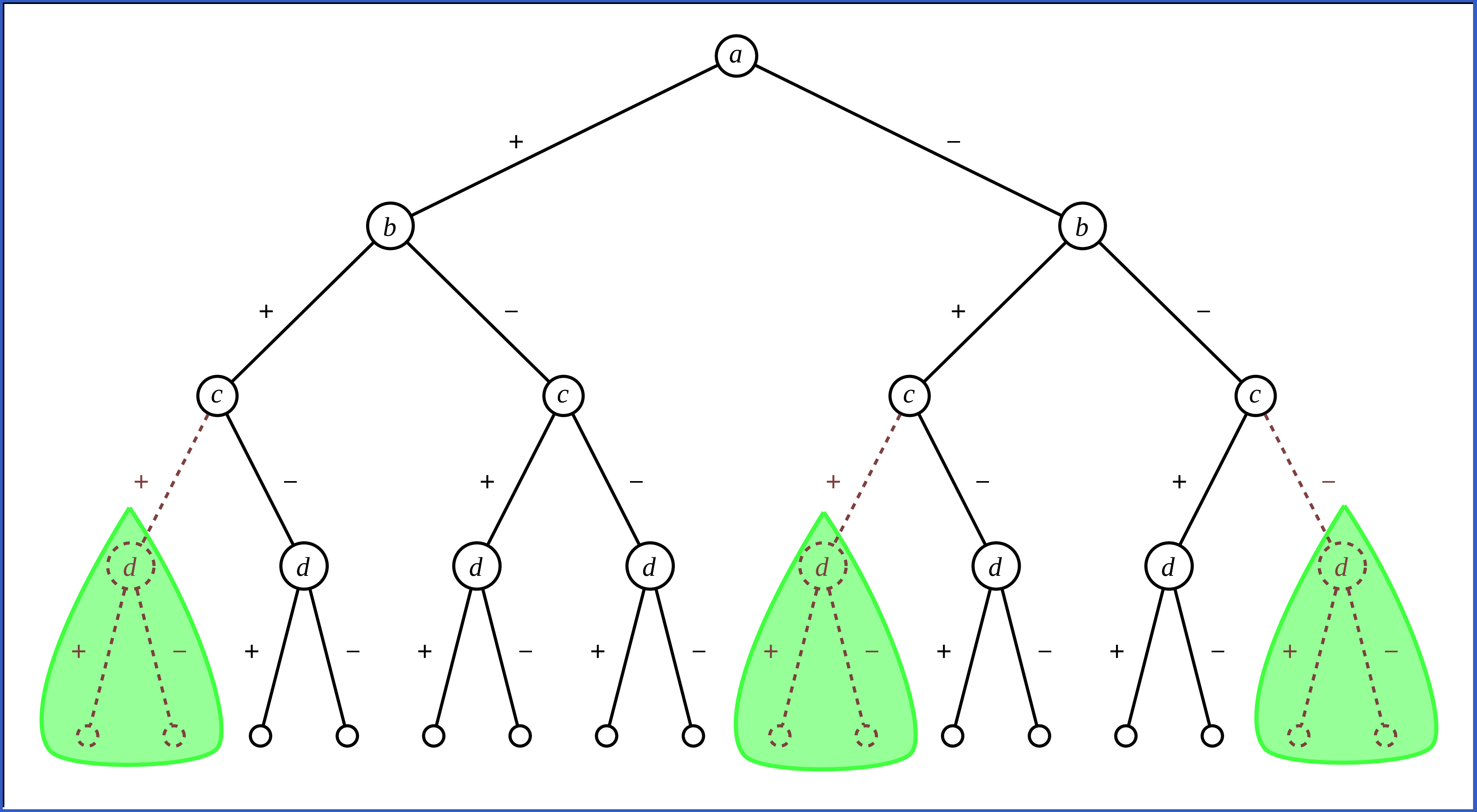
We know that:

- $X' = (\neg c)$

It is obvious that we must assign c false:

- $A' = A \cup \{c \mapsto F\} = \{a \mapsto F,\ \ b \mapsto T,\ \ c \mapsto F\}$
- $X'' \mapsto T$

Effect of unit propagation



Unit propagation

During search, if only a single literal remains in a clause, assign that literal true.

- Most important inference rule used in solving SAT, and part of the original Davis-Putnam procedure
- Typically, unit propagations outnumber branches by about 3:1
- Branches are guesses, while unit propagations are deductions

Evaluating unit propagation

- Unit propagation is an extremely worthwhile inference technique to add to a search procedure
  - Substantial pruning
  - Simple reasoning
- Implementation details have a massive impact on performance, even for a simple and effective technique such as unit propagation

Evaluating inference techniques

- When examining the potential worth of an inference technique, we need to compare:
  - How much of the search space will be pruned (the benefit)
  - How much time and space executing the technique will require (the cost)
- Some of this comparison can be done theoretically
- Implementation details cannot be avoided

Example of resolution

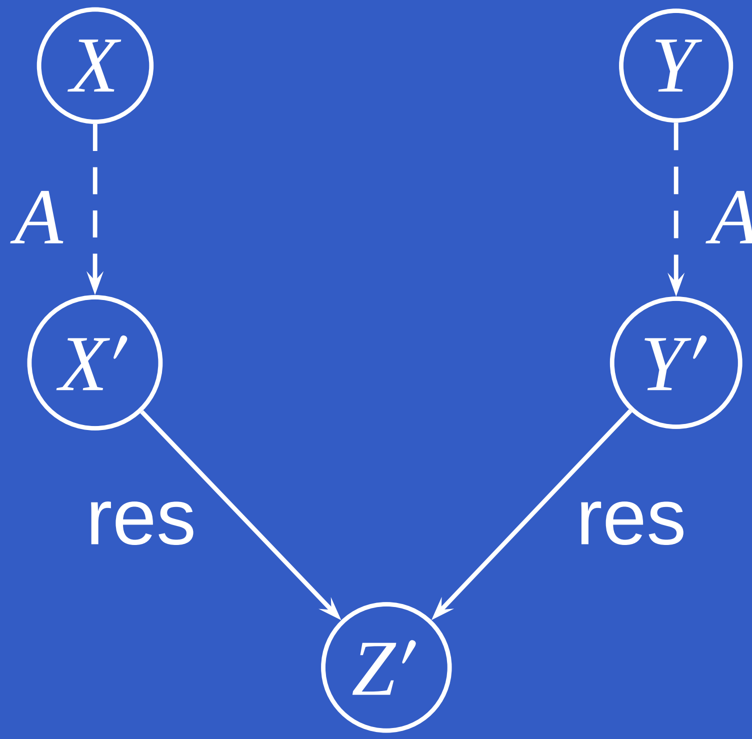
$$\frac{\begin{array}{ccccccccc} & & a & \vee & c & \vee & \boxed{d} \\ & & \neg b & \vee & c & \vee & \boxed{\neg d} \end{array}}{a \vee \neg b \vee c}$$

Example of neighbour resolution

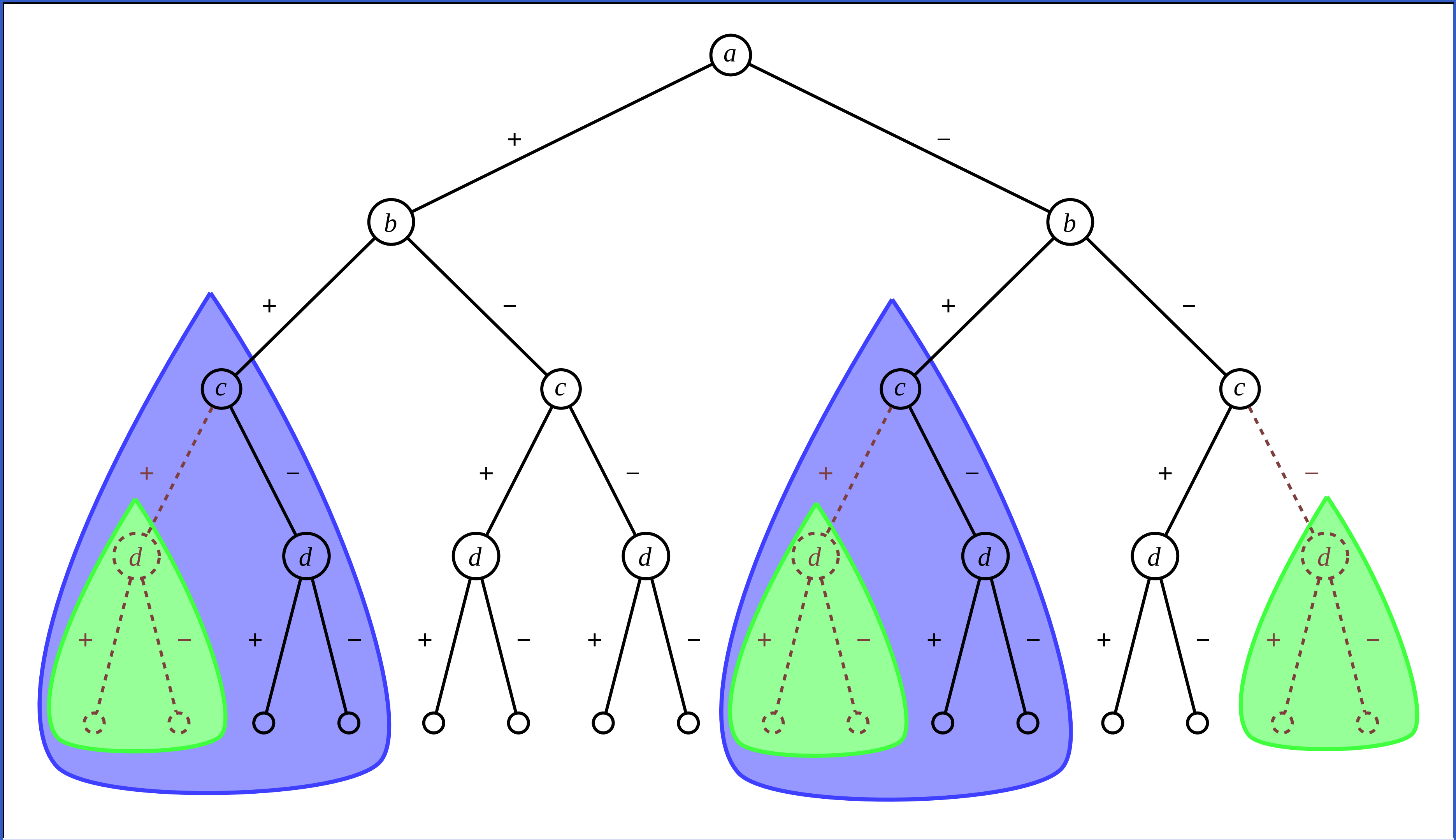
- $X = (\neg a \vee \neg b \vee \neg c)$
- $Y = (a \vee \neg b \vee \neg c)$

$$\begin{array}{ccccc} \boxed{\neg a} & \vee & \neg b & \vee & \neg c \\ \boxed{a} & \vee & \neg b & \vee & \neg c \\ \hline & & \neg b & \vee & \neg c \end{array}$$

Neighbour resolution



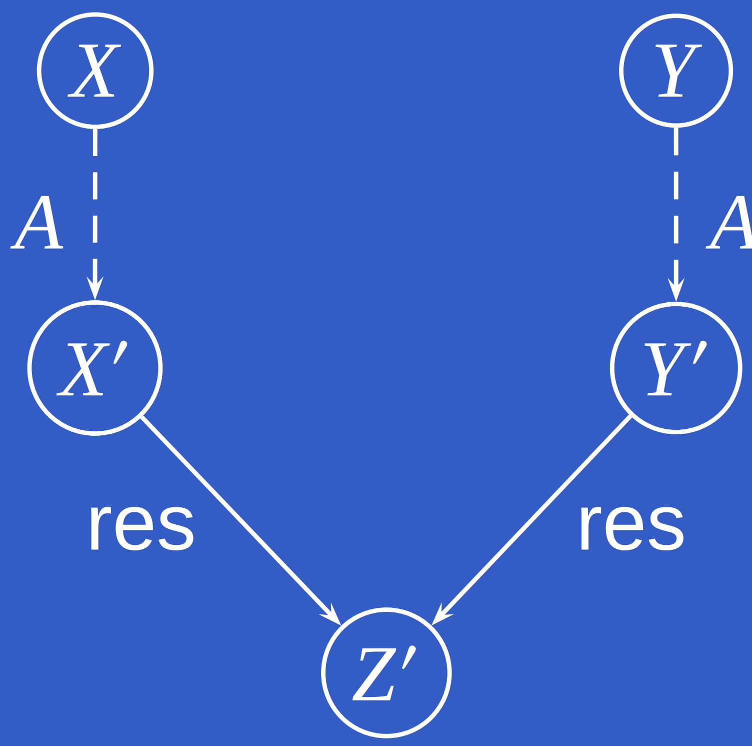
Effect of neighbour resolution



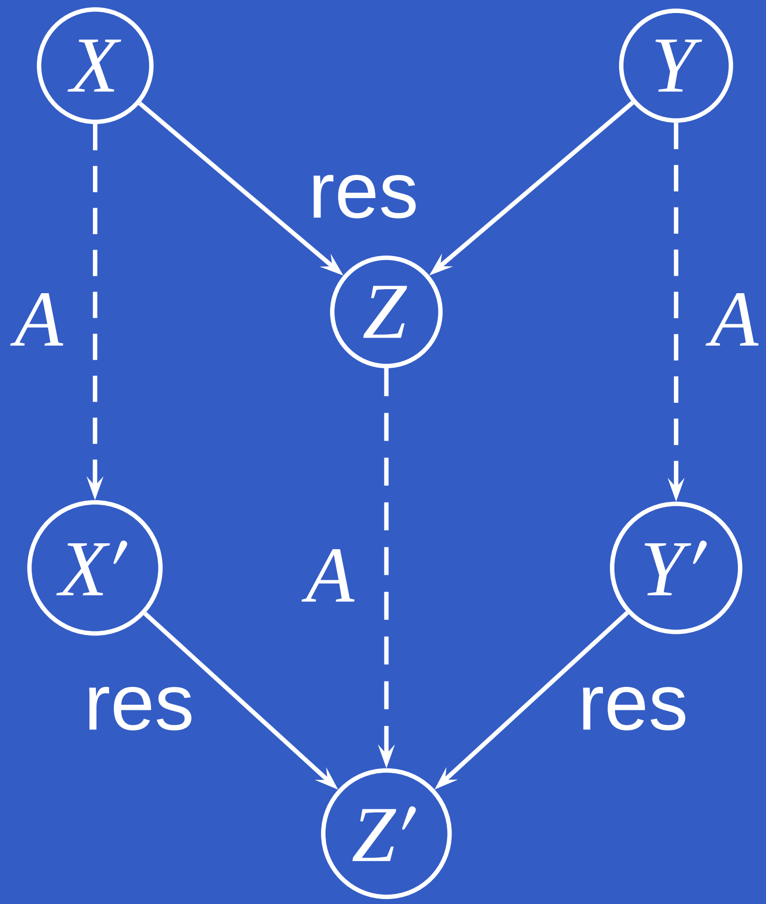
Evaluating neighbour resolution

- Neighbour resolution during search significantly prunes the search space on many problems
- Identifying neighbouring clauses during search takes a great deal of time
- The time cost outweighs the benefit, meaning that this implementation of neighbour resolution during search is not practically beneficial

Neighbour resolution revisited



Preprocessing neighbour resolution

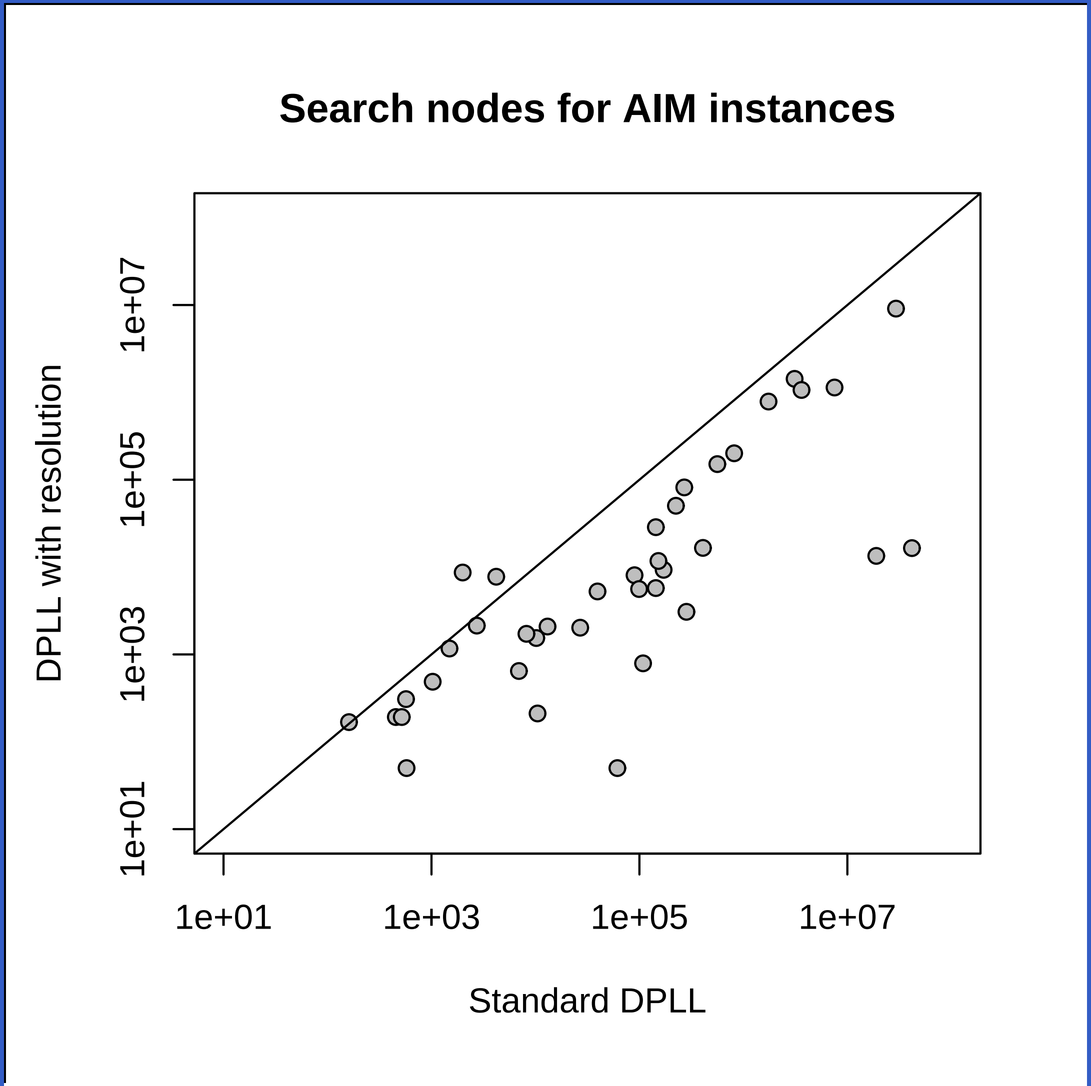


Preprocessing neighbour resolution

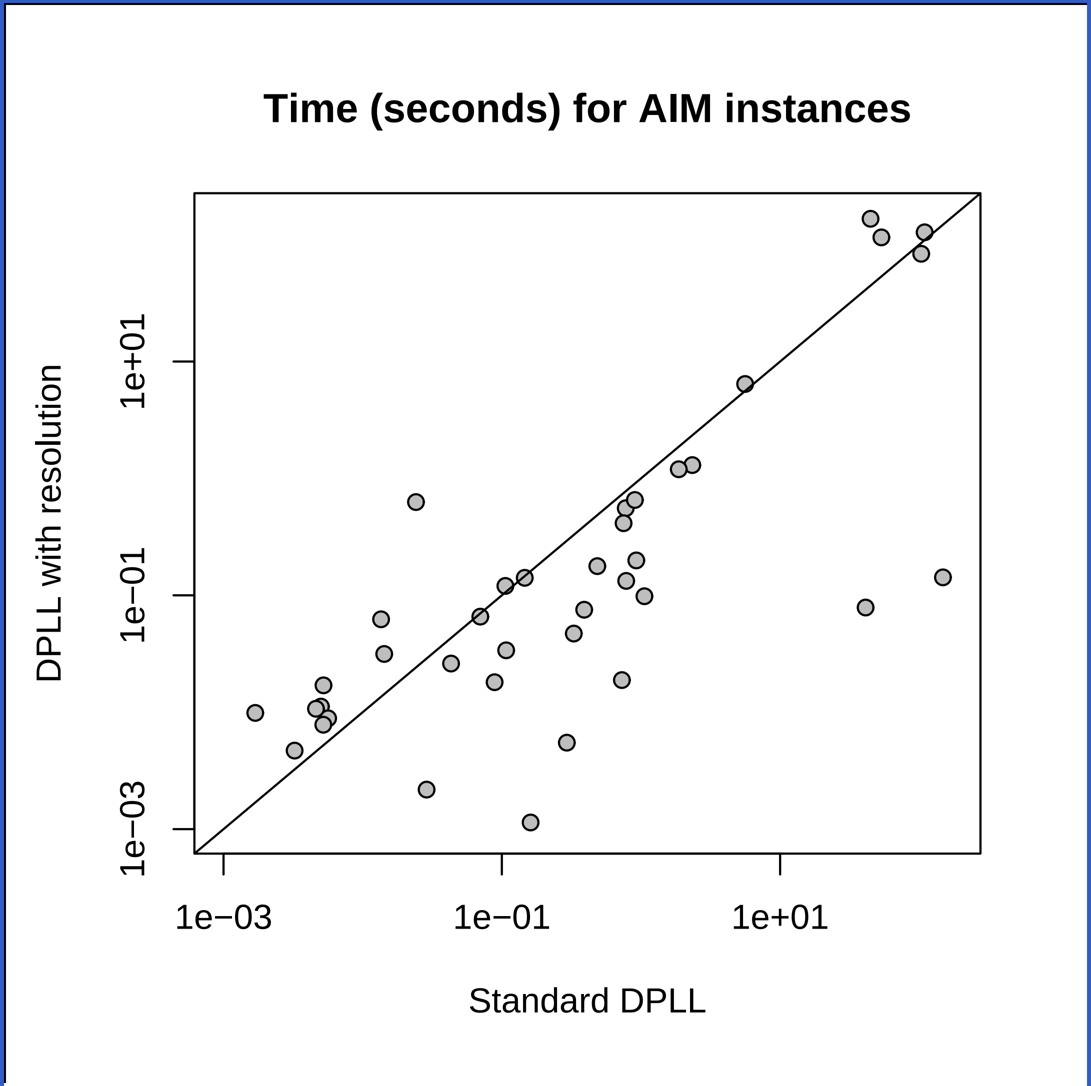
$$\begin{array}{r} a \lor c \lor \boxed{d} \\ \neg b \lor c \lor \boxed{\neg d} \\ \hline a \lor \neg b \lor c \end{array}$$

- $Z = (a \lor \neg b \lor c)$
- $A = \{a \mapsto F,\ \ b \mapsto T\}$
- $Z' = (c)$

Evaluating preprocessing NR



Evaluating preprocessing NR



Future work

- Complete neighbour resolution work
- More theoretical evaluation
  - How much pruning?
  - How much work compared to unit propagation?
- Efficiently combining nogood recording with other inference techniques
- Hyper-resolution
- Look at preprocessing techniques

Conclusions

- SAT is both theoretically interesting and practically important
- Inference can successfully augment search on SAT problems
- The challenge is to find inference techniques that are cost-effective

Related work

Combining resolution and search:

- Rish and Dechter. Resolution versus search: two strategies for SAT. In SAT2000, IOS Press, 2000.
- van Gelder. Satisfiability testing with more reasoning and less guessing. In Second DIMACS implementation challenge, 1995.
- Cha and Iwama. Adding new clauses for faster local search. In Proc AAAI-96, 1996.

Other worthwhile inference techniques

- Nogood recording
- Conflict-directed backjumping
- Equivalency reasoning
- Restrictions of resolution
- Hyper-resolution
- Variable probing

Future work: improved simulation

In our current implementation of simulated neighbour resolution:

- Subsumption during search is ignored
  - We can mark resolvent clauses and cheaply apply subsumption to just those clauses during search
- Extra resolvents (not corresponding to actual neighbour resolvents) are added
  - We can use knowledge of the branching heuristic to determine which resolvents correspond to actual neighbour resolvents

Future work: improved implementation

- Neighbour resolution during search is slow because identifying neighbouring clauses is expensive
  - We have an improved algorithm for neighbour identification which we plan to implement
- It is not worth applying resolution to some problem classes (e.g. the JNH SATLIB instances)
  - We are developing syntactic methods for identifying some such problem classes

Future work: investigation

- If implied clauses are visible to the branching heuristic, the search tree may actually be grow instead of being pruned
  - We plan to investigate the effect of including the implied clauses, but making the branching heuristic ignore them

My other work

Investigating preprocessing techniques:

- Systematic comparison of existing techniques
- Selecting and evaluating novel techniques
  - E.g. taking first-order techniques and applying them to SAT problems