Boolean Satisfiability and SAT Solvers

SAV, March 18th, 2015

Boolean Satisfiability

Boolean Satisfiability

- Goal: find a model that satisfies a propositional formula.

Boolean Satisfiability

- Goal: find a model that satisfies a propositional formula.

$$a \wedge (\neg b \vee c)$$

Boolean Satisfiability

- Goal: find a model that satisfies a propositional formula.

$$a \land (\neg b \lor c) \longrightarrow a \mapsto T,\ b \mapsto F,\ c \mapsto F$$

Boolean Satisfiability

- Goal: find a model that satisfies a propositional formula.

$a \wedge (\neg b \vee c) \longrightarrow a \mapsto T, b \mapsto F, c \mapsto F$

$a \wedge b \wedge (\neg b \vee \neg a)$

Boolean Satisfiability

- Goal: find a model that satisfies a propositional formula.

$a \wedge (\neg b \vee c) \longrightarrow a \mapsto T, b \mapsto F, c \mapsto F$

$a \wedge b \wedge (\neg b \vee \neg a) \longrightarrow$ unsatisfiable

Boolean Satisfiability

- Goal: find a model that satisfies a propositional formula.

$a \wedge (\neg b \vee c) \longrightarrow a \mapsto T, b \mapsto F, c \mapsto F$

$a \wedge b \wedge (\neg b \vee \neg a) \longrightarrow$ unsatisfiable

- The original NP-complete problem.
  - "As hard as any other problem in NP."

Boolean Satisfiability

- Goal: find a model that satisfies a propositional formula.

$$a \wedge (\neg b \vee c) \longrightarrow a \mapsto T, b \mapsto F, c \mapsto F$$

$$a \wedge b \wedge (\neg b \vee \neg a) \longrightarrow \text{unsatisfiable}$$

- The original NP-complete problem.
  - "As hard as any other problem in NP."
- S. Cook, *The complexity of theorem proving procedures*, STOC 1971.

SAT in Practice

- Ubiquitous in hardware/circuit design
  - E.g. equivalence checking.
- Search/AI problems
  - E.g. reduce Sudoku to SAT.
  - Dependency management in Eclipse.
- Software verification
  - By itself, and as part of the SMT stack.

Decidability is Trivial

- For *n* variables, enumerate all 2^n possible assignments.

Decidability is Trivial

- For *n* variables, enumerate all 2^n possible assignments.

$\varphi \equiv a \land (\neg b \lor c)$

a	b	c	φ
T	T	T	T
T	T	F	F
T	F	T	T
T	F	F	T
F	T	T	F
F	T	F	F
F	F	T	F
F	F	F	F

Decidability is Trivial

- For n variables, enumerate all 2^n possible assignments.

$\varphi \equiv a \land (\neg b \lor c)$

a	b	c	φ
T	T	T	T
T	T	F	F
T	F	T	T
T	F	F	T
F	T	T	F
F	T	F	F
F	F	T	F
F	F	F	F

- Obviously not very efficient. SAT solving is all about making this enumeration “smart”.

Going to Clauses

- SAT solving (almost) always applies to formulas normalized to *conjunctive normal form* (CNF).

Going to Clauses

- SAT solving (almost) always applies to formulas normalized to *conjunctive normal form* (CNF).

$$(a \lor \neg b \lor c) \land (\neg a \lor c \lor d \lor \neg e) \land (b \lor \neg d \lor e)$$

Going to Clauses

- SAT solving (almost) always applies to formulas normalized to *conjunctive normal form* (CNF).

$(a \lor \neg b \lor c) \land (\neg a \lor c \lor d \lor \neg e) \land (b \lor \neg d \lor e)$

$\{ \{ a, \overline{b}, c \}, \{ \overline{a}, c, d, \overline{e} \}, \{ b, \overline{d}, e \} \}$

Going to Clauses

- SAT solving (almost) always applies to formulas normalized to *conjunctive normal form* (CNF).

$(a \lor \neg b \lor c) \land (\neg a \lor c \lor d \lor \neg e) \land (b \lor \neg d \lor e)$

$\{ \{ a, \overline{b}, c \}, \{ \overline{a}, c, d, \overline{e} \}, \{ b, \overline{d}, e \} \}$

$(a + \overline{b} + c)(\overline{a} + c + d + \overline{e})(b + \overline{d} + e)$

Going to Clauses

- SAT solving (almost) always applies to formulas normalized to *conjunctive normal form* (CNF).

$$(a \lor \neg b \lor c) \land (\neg a \lor c \lor d \lor \neg e) \land (b \lor \neg d \lor e)$$

$$\{ \{ a, \overline{b}, c \}, \{ \overline{a}, c, d, \overline{e} \}, \{ b, \overline{d}, e \} \}$$

$$(a + \overline{b} + c)(\overline{a} + c + d + \overline{e})(b + \overline{d} + e)$$

Note that a truth table is a kind of *disjunctive* normal form (DNF).

First Approach: Resolution

a ∨ ¬b ∨ f

¬a ∨ ¬c ∨ d ∨ ¬e

First Approach: Resolution

a ∨ ¬b ∨ f

¬a ∨ ¬c ∨ d ∨ ¬e

First Approach: Resolution

$a \lor \neg b \lor f$

$\neg a \lor \neg c \lor d \lor \neg e$

$\neg b \lor f \lor \neg c \lor d \lor \neg e$

First Approach: Resolution

a ∨ ¬b ∨ f   ¬a ∨ ¬c ∨ d ∨ ¬e

¬b ∨ f ∨ ¬c ∨ d ∨ ¬e

- Resolution eliminates one variable by producing a new clause (*resolvent*) from complementary ones.

Resolution

$$(a \lor b) \land (a \lor \neg b) \land (\neg a \lor c) \land (\neg a \lor \neg c)$$

Resolution

$(a \lor b) \land (a \lor \neg b) \land (\neg a \lor c) \land (\neg a \lor \neg c)$

Resolution

$$(a \lor b) \land (a \lor \neg b) \land (\neg a \lor c) \land (\neg a \lor \neg c)$$

$$a \land (\neg a \lor c) \land (\neg a \lor \neg c)$$

Resolution

$$(a \lor b) \land (a \lor \neg b) \land (\neg a \lor c) \land (\neg a \lor \neg c)$$

$$a \land (\neg a \lor c) \land (\neg a \lor \neg c)$$

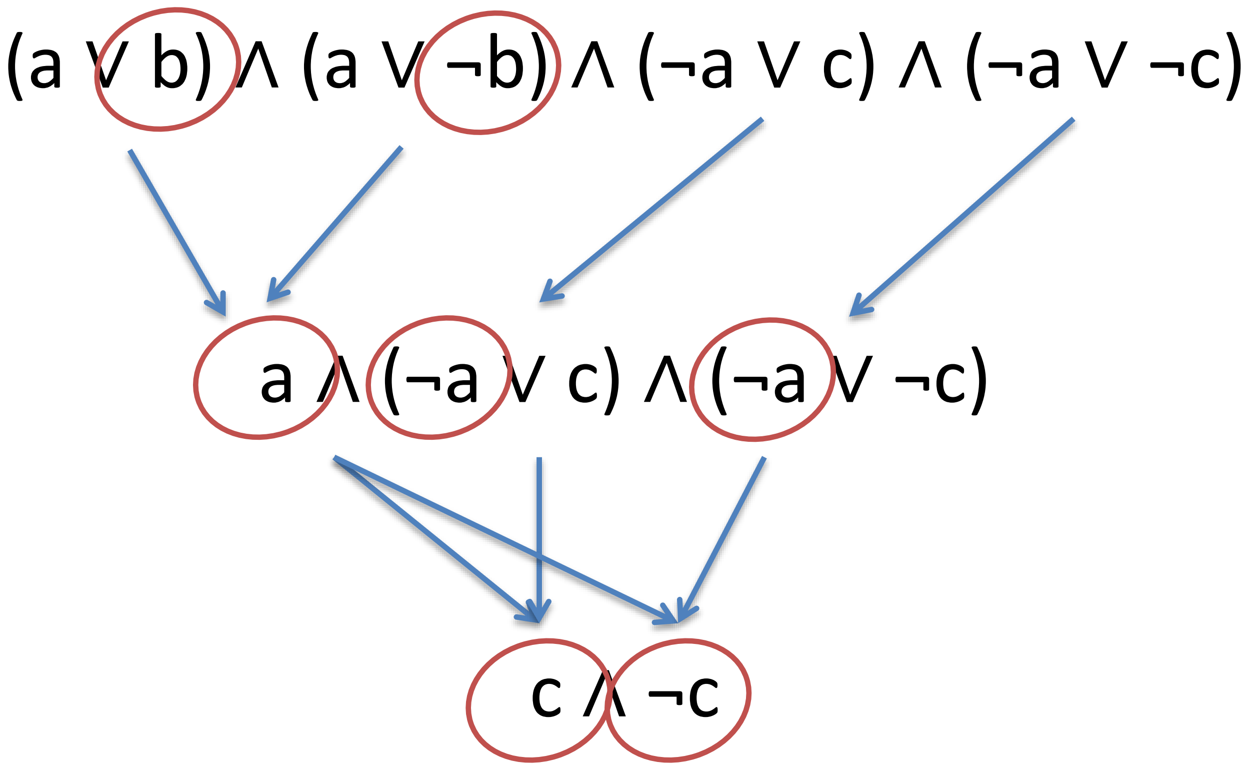
Resolution

$$(a \lor b) \land (a \lor \neg b) \land (\neg a \lor c) \land (\neg a \lor \neg c)$$

$$a \land (\neg a \lor c) \land (\neg a \lor \neg c)$$

$$c \land \neg c$$

Resolution



(Part of) Davis Putnam Algorithm

- (Also: when a variable appears in only one polarity, remove all clauses containing it.)

- M. Davis, H. Putnam, *A computing procedure for quantification theory*, JACM, 1960.

- Problem: space explosion!

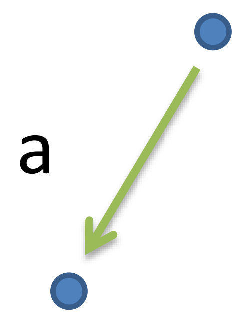
- DP is *proof-oriented*. Current algorithms are *model-oriented*.

Backtracking Search

$$
\begin{aligned}
& (\qquad\quad\ b \lor \neg c) \\
\land\ & (\neg a \lor b \lor c) \\
\land\ & (\neg a \lor \neg b)
\end{aligned}
$$

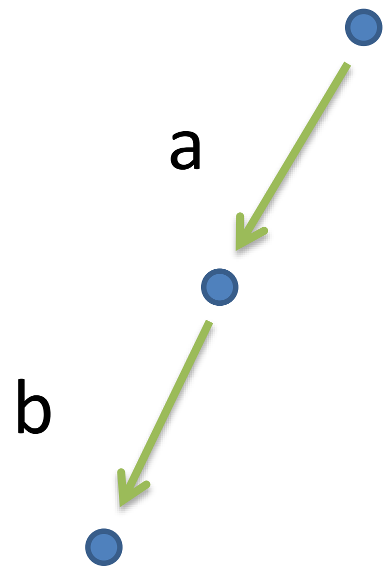
Backtracking Search

$$
\begin{aligned}
& (b \lor \neg c) \\
\land\ & (\neg a \lor b \lor c) \\
\land\ & (\neg a \lor \neg b)
\end{aligned}
$$



Backtracking Search

$$
\begin{aligned}
& (\qquad\qquad b \lor \neg c) \\
\land\ & (\neg a \lor b \lor c) \\
\land\ & (\neg a \lor \neg b \qquad)
\end{aligned}
$$

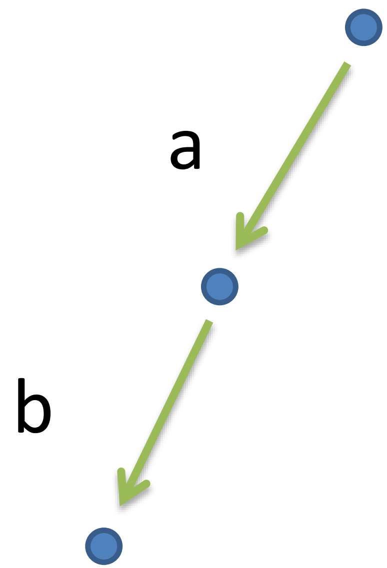


Backtracking Search

$(\quad b \lor \neg c)$

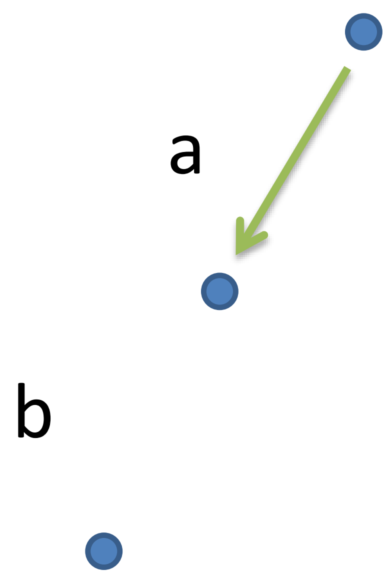
$\land (\neg a \lor b \lor c)$

$\land (\neg a \lor \neg b)$!



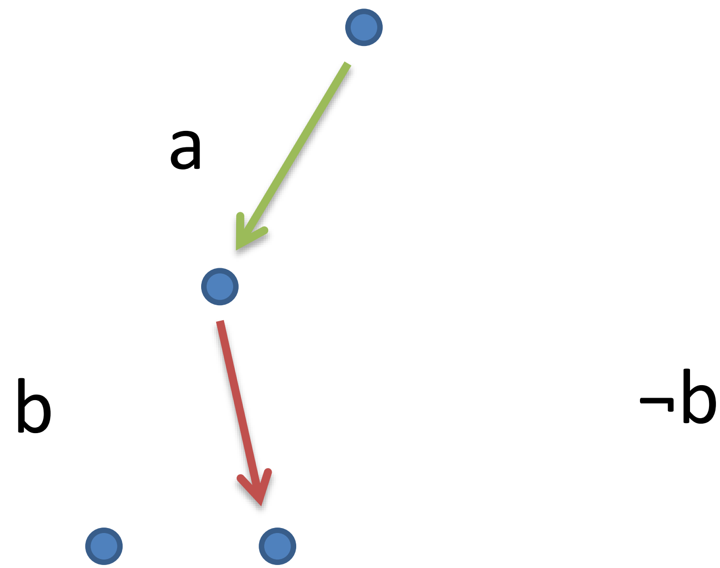
Backtracking Search

$$(\quad b \lor \neg c)$$
$$\land (\neg a \lor b \lor c)$$
$$\land (\neg a \lor \neg b \quad)$$



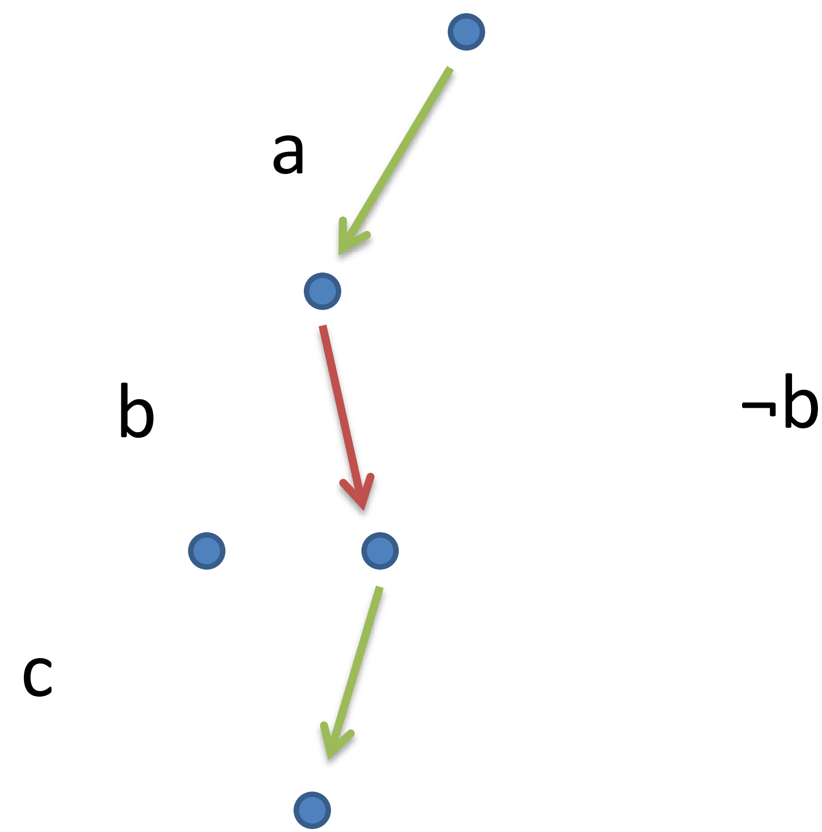
Backtracking Search

$$
\begin{aligned}
& (\; b \lor \neg c \;) \\
\land\; & (\; \neg a \lor b \lor c \;) \\
\land\; & (\; \neg a \lor \neg b \;)
\end{aligned}
$$



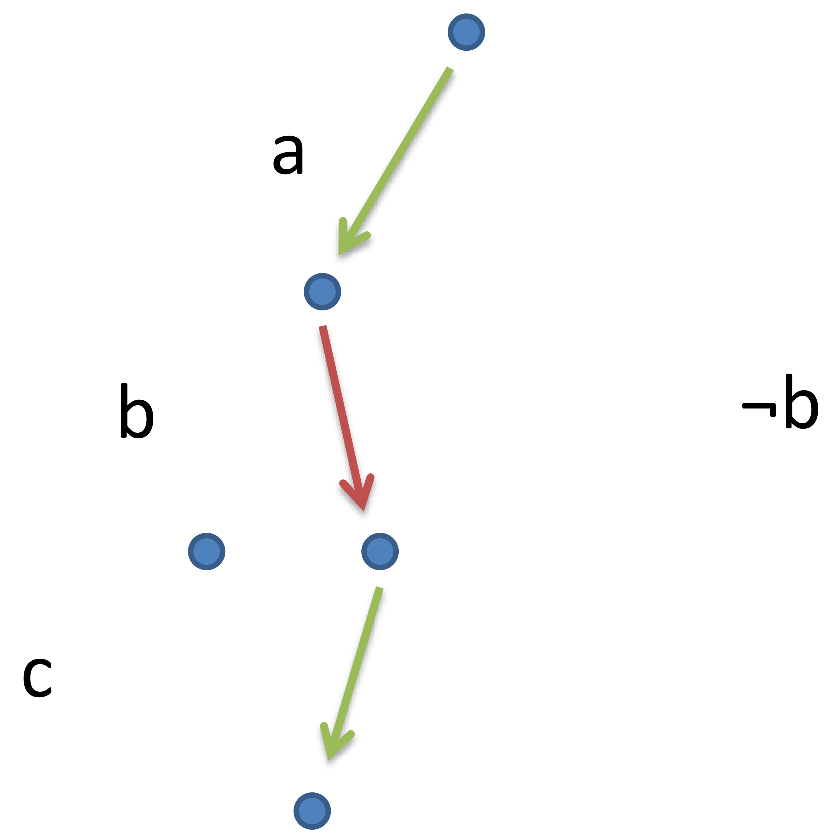
Backtracking Search

$$
\begin{aligned}
& (\quad\quad\quad\; b \lor \neg c\,) \\
\land\; & (\neg a \lor b \lor \;\;c\,) \\
\land\; & (\neg a \lor \neg b \quad\quad\;\;)
\end{aligned}
$$



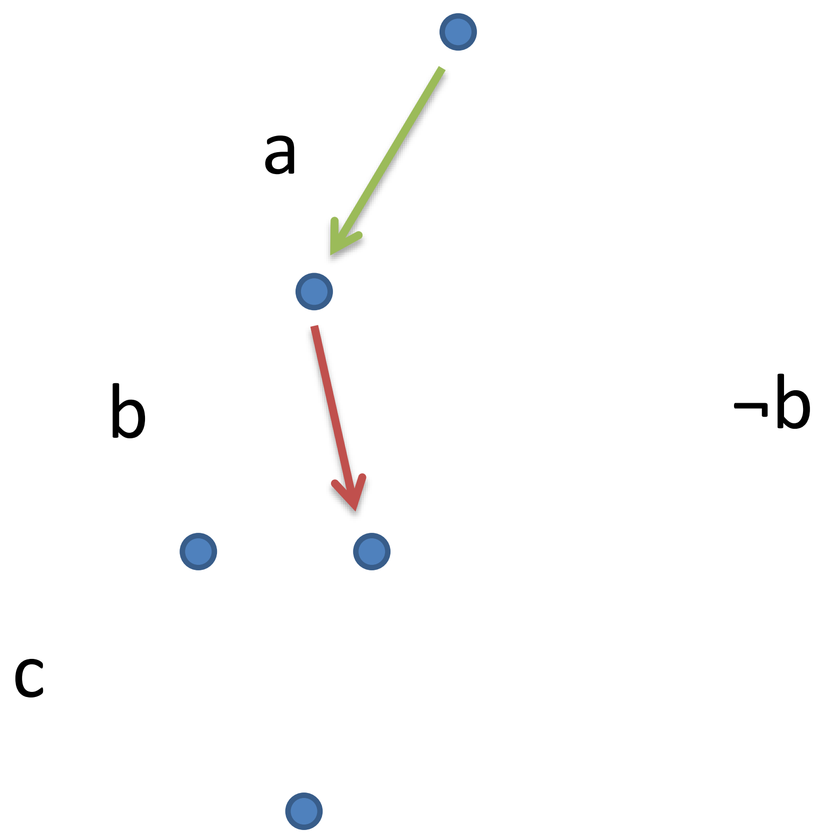
Backtracking Search

$$(b \lor \neg c) \; !$$
$$\land (\neg a \lor b \lor c)$$
$$\land (\neg a \lor \neg b)$$



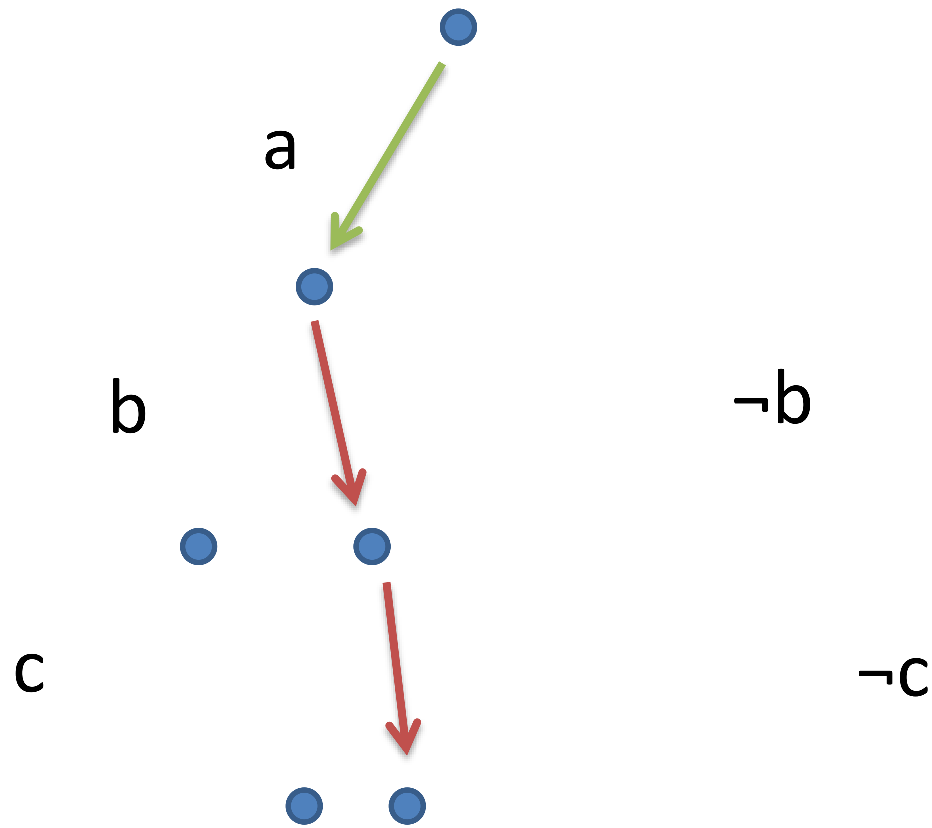
Backtracking Search

$$
\begin{aligned}
&(\quad b \lor \neg c\,) \\
\land\, &(\,\neg a \lor b \lor c\,) \\
\land\, &(\,\neg a \lor \neg b\,)
\end{aligned}
$$



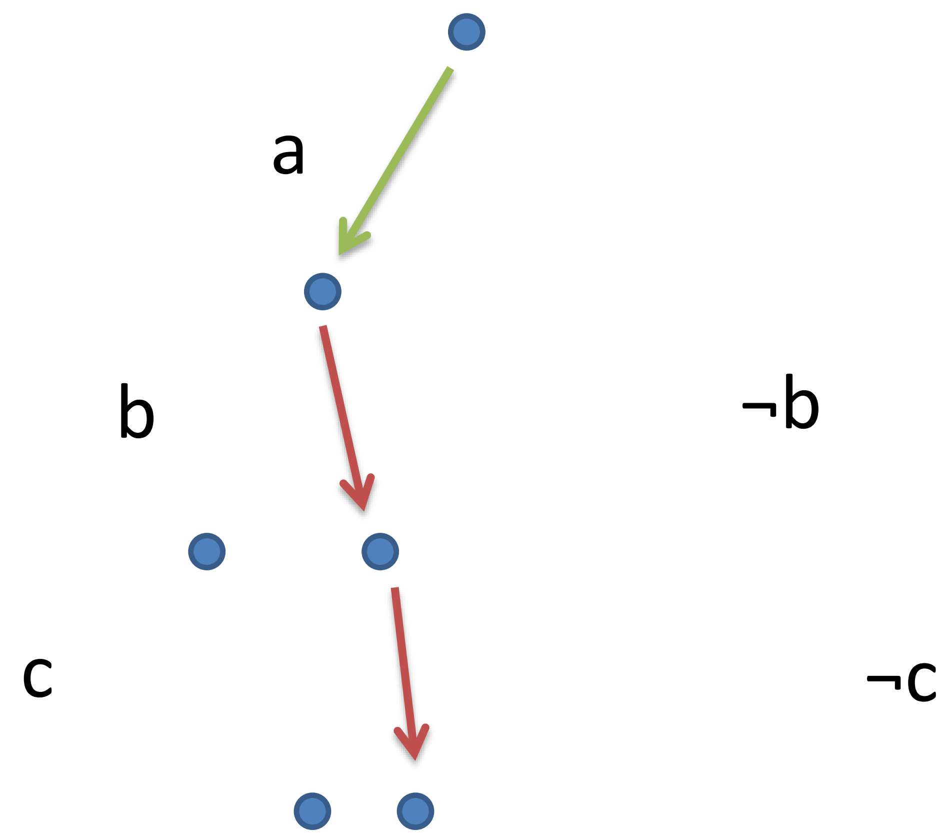
Backtracking Search

$$
\begin{aligned}
&(\quad\ \ b \lor \neg c)\\
\land\ &(\neg a \lor b \lor c)\\
\land\ &(\neg a \lor \neg b\quad\ \ )
\end{aligned}
$$



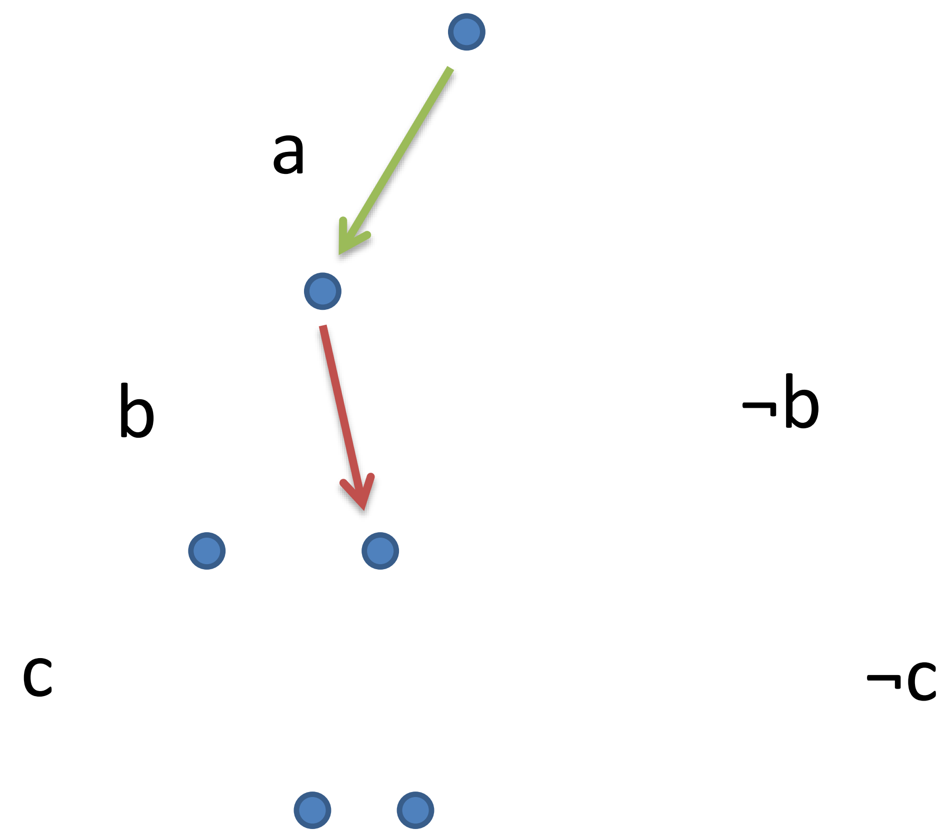
Backtracking Search

$$
\begin{aligned}
&(\qquad\quad b \lor \neg c\,) \\
\land\, &(\neg a \lor b \lor c\,)\ \textbf{!} \\
\land\, &(\neg a \lor \neg b \qquad)
\end{aligned}
$$



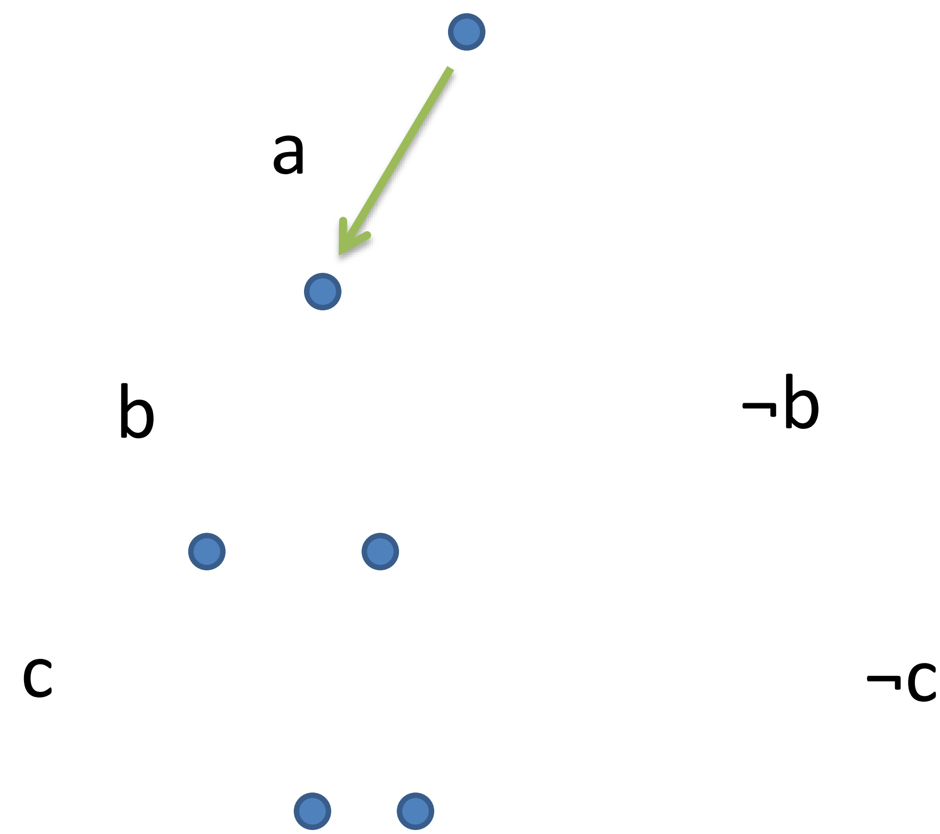
Backtracking Search

$$
\begin{aligned}
&(\phantom{\neg a \lor{}} b \lor \neg c) \\
\land\ &(\neg a \lor b \lor c) \\
\land\ &(\neg a \lor \neg b)
\end{aligned}
$$



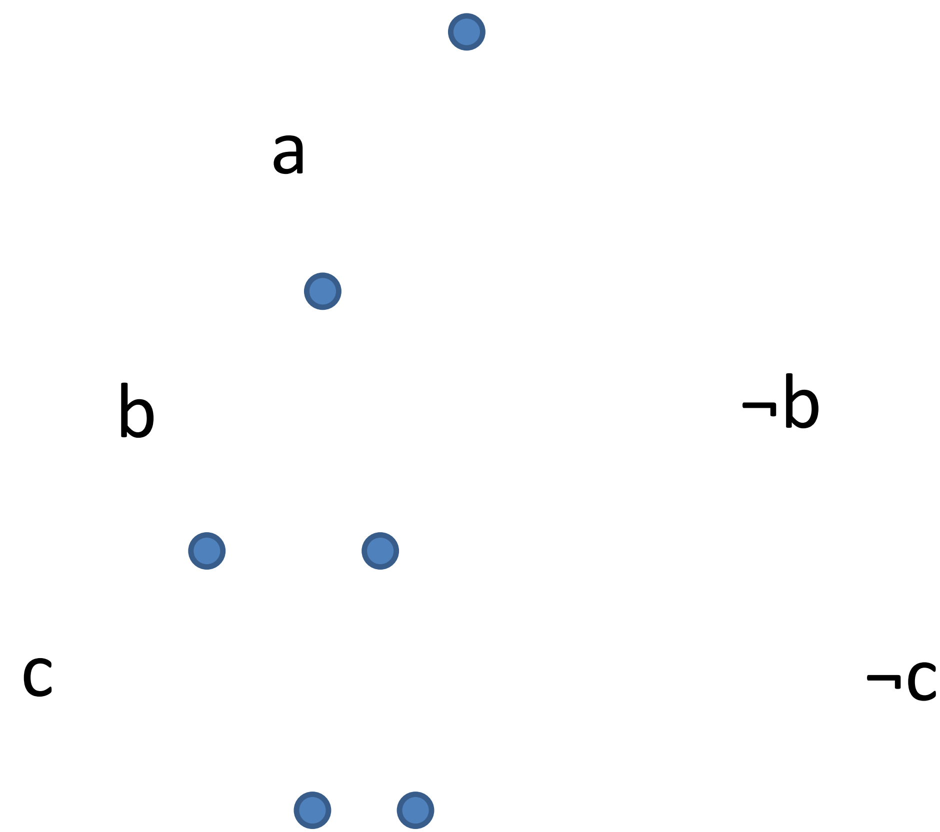
Backtracking Search

$$
\begin{aligned}
& (b \lor \neg c) \\
\land\ & (\neg a \lor b \lor c) \\
\land\ & (\neg a \lor \neg b)
\end{aligned}
$$



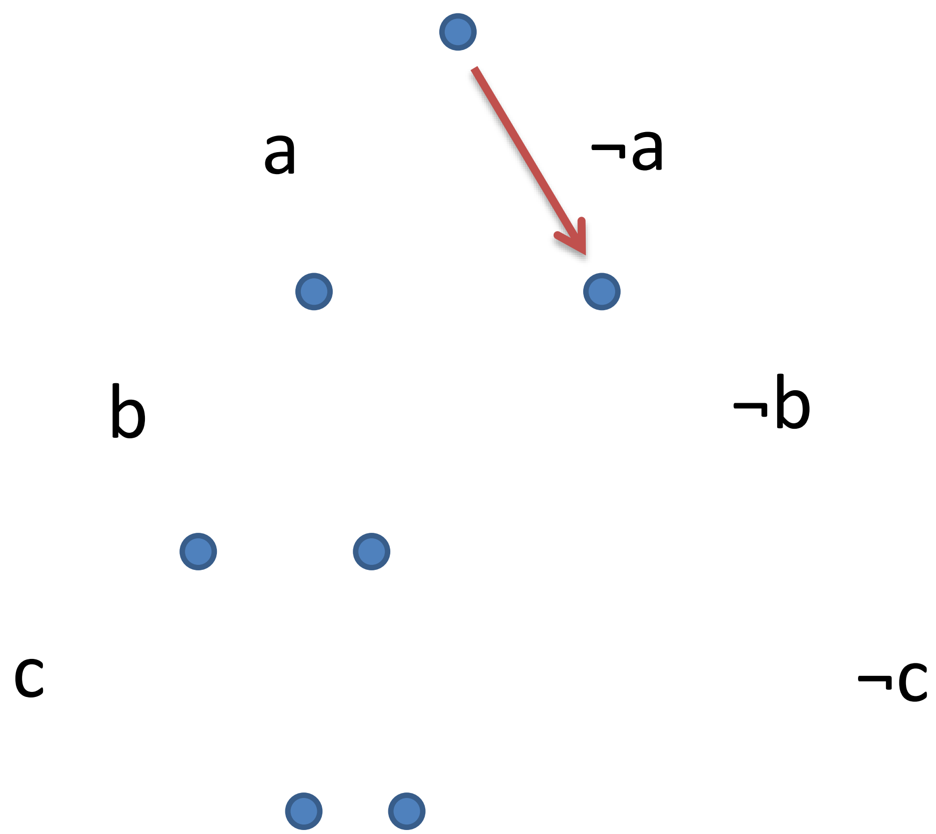
Backtracking Search

$$
\begin{aligned}
&(\quad\quad\; b \lor \neg c\,) \\
\land\, &(\neg a \lor b \lor c\,) \\
\land\, &(\neg a \lor \neg b\quad\;)
\end{aligned}
$$



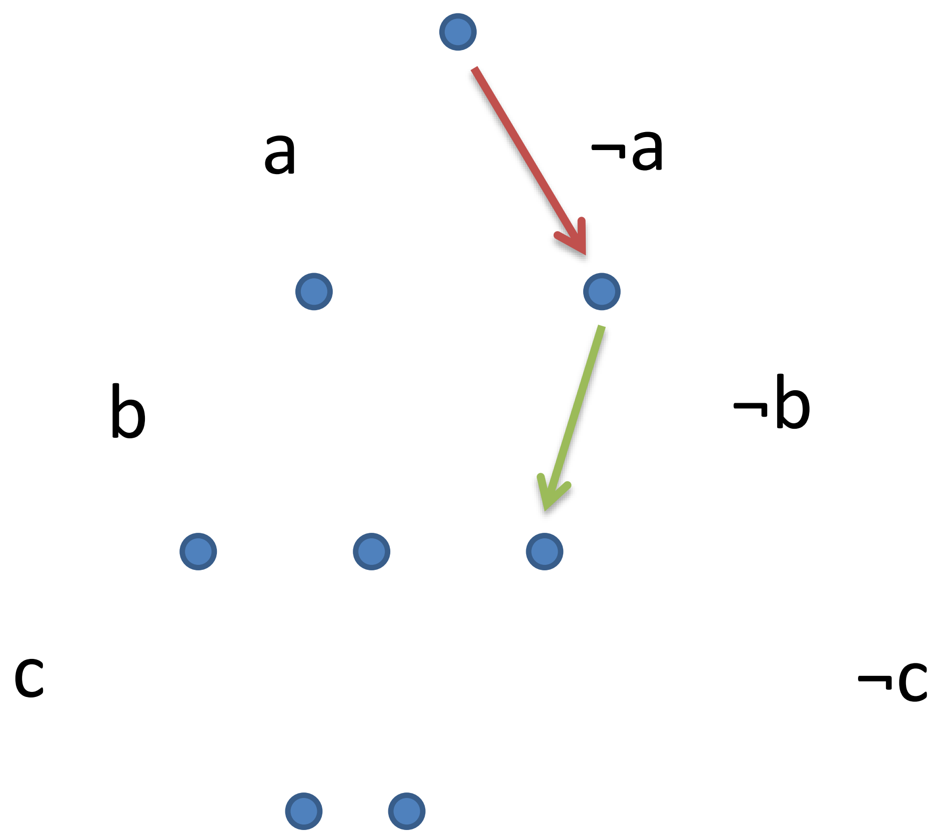
Backtracking Search

$$
\begin{aligned}
&(\quad b \lor \neg c) \\
\land &(\neg a \lor b \lor c) \\
\land &(\neg a \lor \neg b \quad)
\end{aligned}
$$



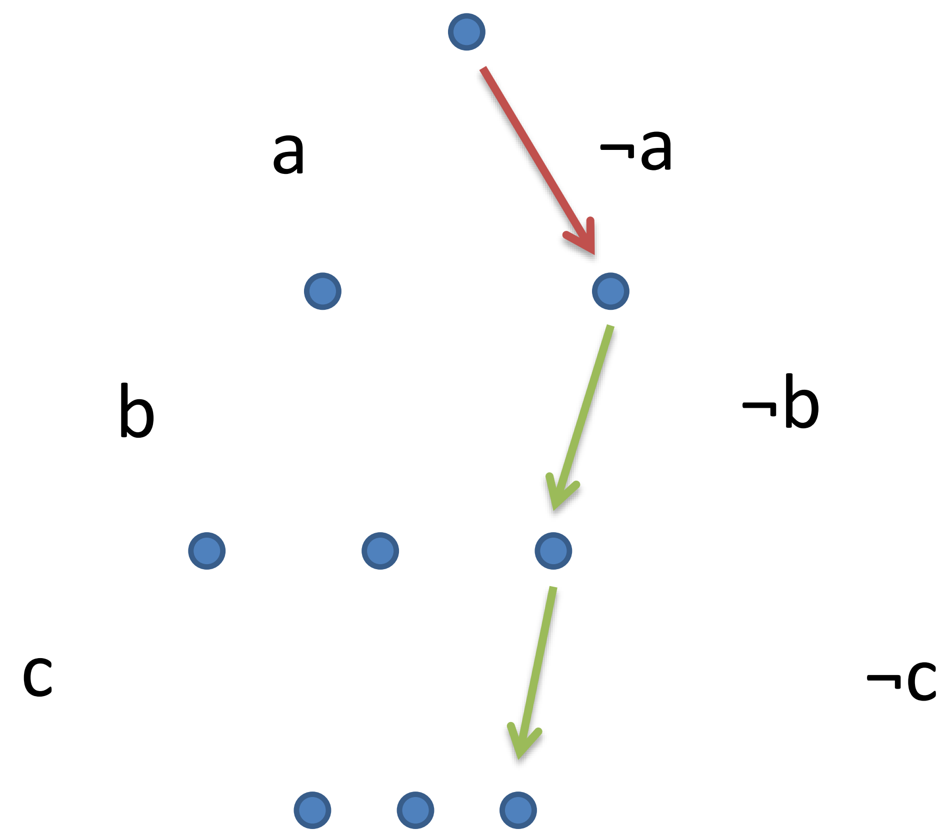
Backtracking Search

$$
\begin{aligned}
&(\phantom{\neg a \vee{}} b \vee \neg c) \\
\wedge\ &(\neg a \vee b \vee c) \\
\wedge\ &(\neg a \vee \neg b)
\end{aligned}
$$



Backtracking Search

$$
\begin{aligned}
& (\; b \lor \neg c \;) \\
\land\; & (\; \neg a \lor b \lor c \;) \\
\land\; & (\; \neg a \lor \neg b \;)
\end{aligned}
$$



Boolean Constraint Propagation

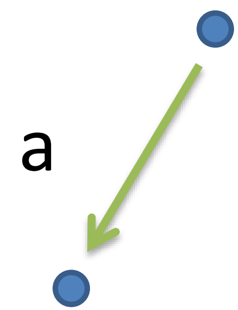
- *"When all but one literal are falsified, it becomes implied."*

$$
\begin{aligned}
& (\quad b \lor \neg c) \\
\land\ & (\neg a \lor b \lor c) \\
\land\ & (\neg a \lor \neg b)
\end{aligned}
$$

Boolean Constraint Propagation

- *"When all but one literal are falsified, it becomes implied."*

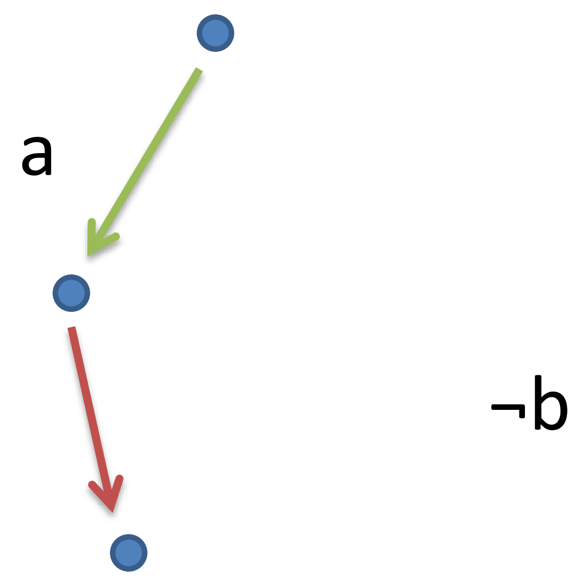
$$
\begin{array}{l}
\quad (\qquad\quad\ b \lor \neg c) \\
\land (\neg a \lor b \lor c) \\
\land (\neg a \lor \neg b \qquad)
\end{array}
$$



Boolean Constraint Propagation

- *"When all but one literal are falsified, it becomes implied."*

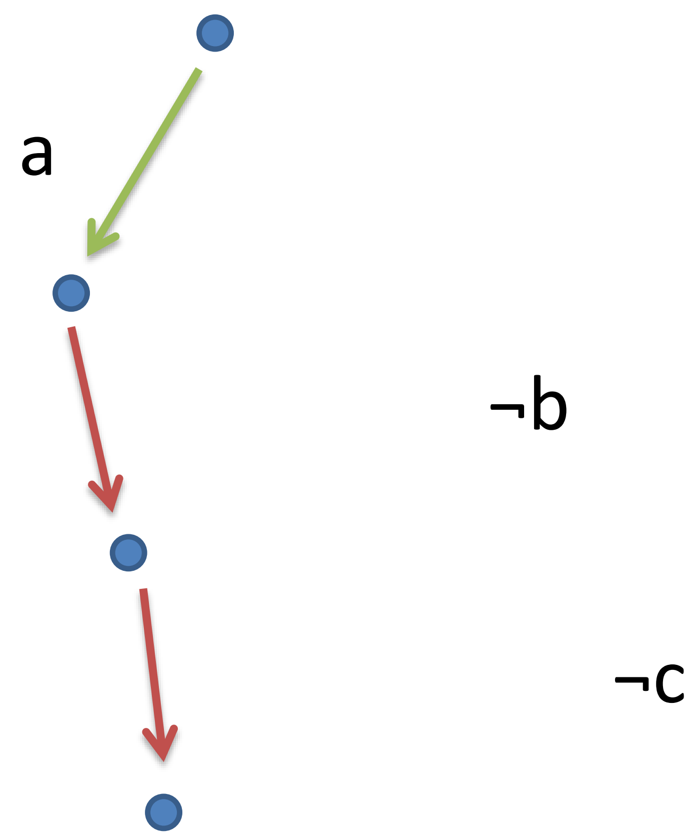
$$
\begin{aligned}
& (\quad\quad\;\; b \lor \neg c\,) \\
\land\; & (\,\neg a \lor b \lor c\,) \\
\land\; & (\,\neg a \lor \neg b \quad\;\;)
\end{aligned}
$$



Boolean Constraint Propagation

- *"When all but one literal are falsified, it becomes implied."*

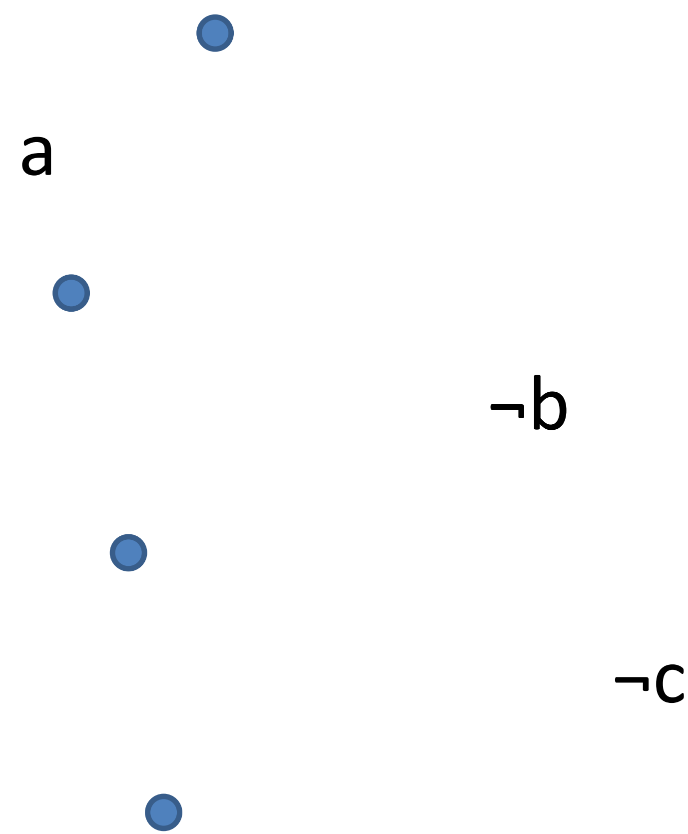
$$
\begin{aligned}
& (b \lor \neg c) \\
\land\ & (\neg a \lor b \lor c) \\
\land\ & (\neg a \lor \neg b)
\end{aligned}
$$



Boolean Constraint Propagation

- *"When all but one literal are falsified, it becomes implied."*

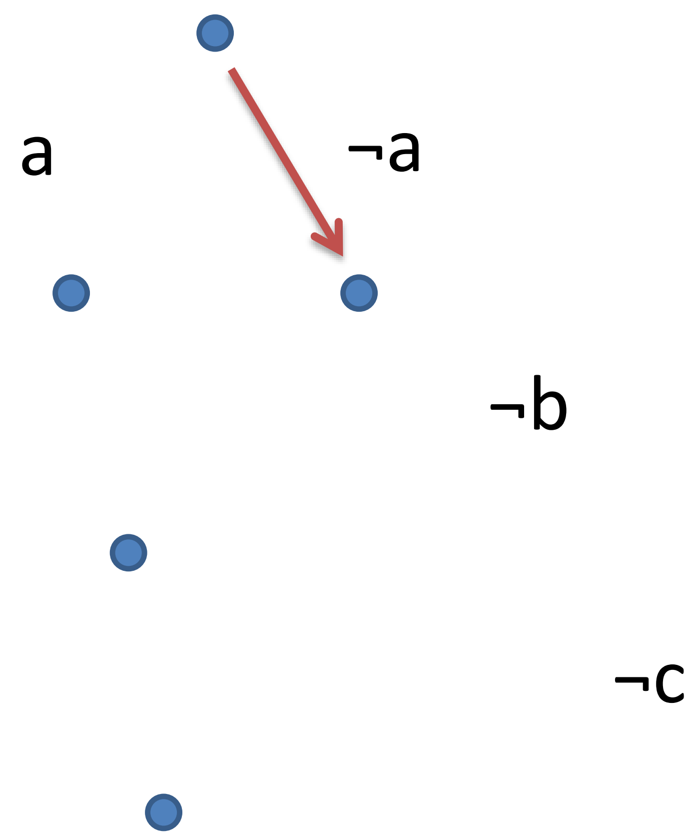
$$
\begin{aligned}
& (b \lor \neg c) \\
\land\ & (\neg a \lor b \lor c) \\
\land\ & (\neg a \lor \neg b)
\end{aligned}
$$



Boolean Constraint Propagation

- *“When all but one literal are falsified, it becomes implied.”*

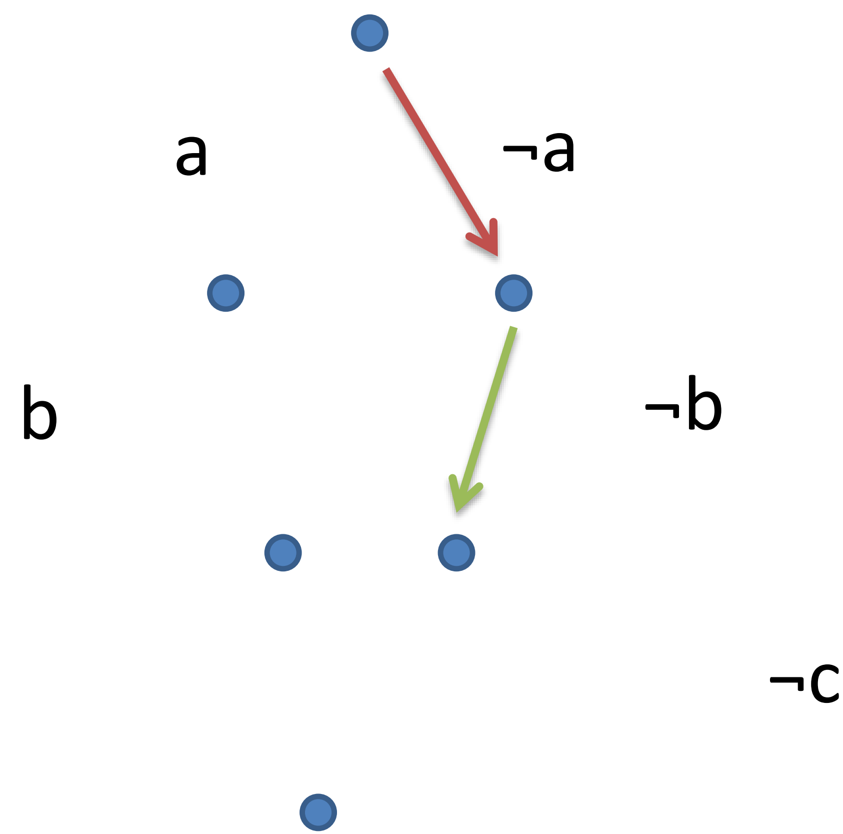
$$
\begin{aligned}
& (\quad b \lor \neg c) \\
\land\ & (\neg a \lor b \lor c) \\
\land\ & (\neg a \lor \neg b \quad)
\end{aligned}
$$



Boolean Constraint Propagation

- *"When all but one literal are falsified, it becomes implied."*

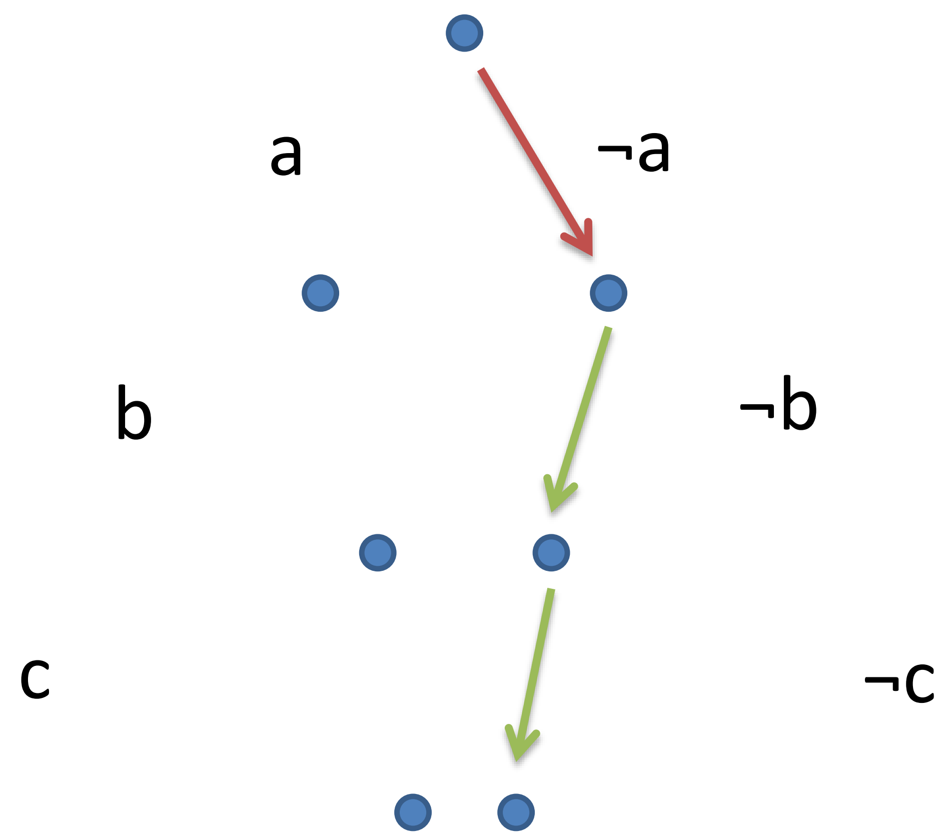
$$(\quad\quad\quad b \lor \neg c)$$
$$\land (\neg a \lor b \lor c)$$
$$\land (\neg a \lor \neg b \quad\quad)$$



Boolean Constraint Propagation

- *"When all but one literal are falsified, it becomes implied."*

$$
\begin{aligned}
& (b \lor \neg c) \\
\land\ & (\neg a \lor b \lor c) \\
\land\ & (\neg a \lor \neg b)
\end{aligned}
$$



Two-watched-literal Scheme for BCP

- BCP can cut the search tree dramatically…
- …but checking each clause for potential implications is expensive.

- Observation: as long as at least two literals in a clause are “not false”, that clause does not imply any new literal.
- Idea: for each clause, try to maintain that invariant.

Cutting Deeper: Learning

- Idea: compute new clauses that are logically implied, and that may trigger more BCP.

- Use an *implication graph*. When a conflict is derived, look for a *small explanation*.

Learning

$(a \lor d)$

$\land\ (a \lor \neg c \lor \neg h)$

$\land\ (a \lor h \lor \neg m)$

$\land\ (b \lor k)$

$\land\ (\neg g \lor \neg c \lor i)$

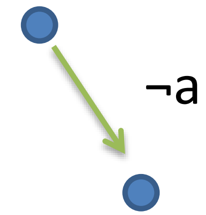
$\land\ (\neg g \lor h \lor \neg i)$

$\land\ (g \lor h \lor \neg j)$

$\land\ (g \lor j \lor \neg m)$

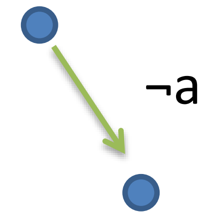
Learning

$$
\begin{aligned}
& (a \lor d) \\
\land\ & (a \lor \neg c \lor \neg h) \\
\land\ & (a \lor h \lor \neg m) \\
\land\ & (b \lor k) \\
\land\ & (\neg g \lor \neg c \lor i) \\
\land\ & (\neg g \lor h \lor \neg i) \\
\land\ & (g \lor h \lor \neg j) \\
\land\ & (g \lor j \lor \neg m)
\end{aligned}
$$



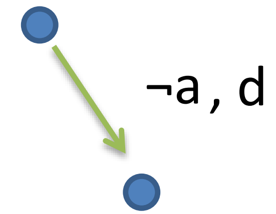
Learning

$$(a \lor d)$$
$$\land\ (a \lor \neg c \lor \neg h)$$
$$\land\ (a \lor h \lor \neg m)$$
$$\land\ (b \lor k)$$
$$\land\ (\neg g \lor \neg c \lor i)$$
$$\land\ (\neg g \lor h \lor \neg i)$$
$$\land\ (g \lor h \lor \neg j)$$
$$\land\ (g \lor j \lor \neg m)$$



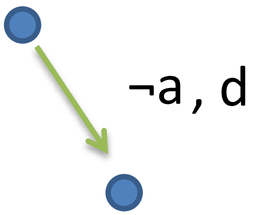
Learning

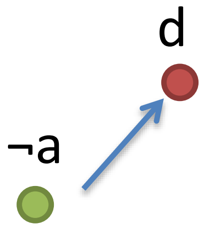
$$
\begin{aligned}
& (a \lor d) \\
\land\ & (a \lor \neg c \lor \neg h) \\
\land\ & (a \lor h \lor \neg m) \\
\land\ & (b \lor k) \\
\land\ & (\neg g \lor \neg c \lor i) \\
\land\ & (\neg g \lor h \lor \neg i) \\
\land\ & (g \lor h \lor \neg j) \\
\land\ & (g \lor j \lor \neg m)
\end{aligned}
$$



Learning

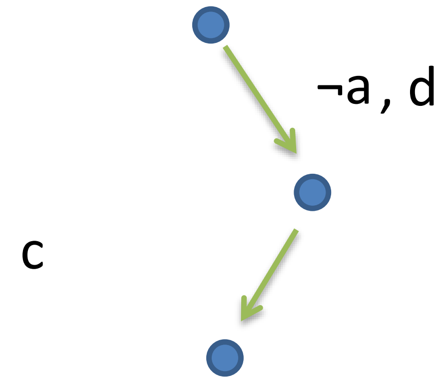
$(a \lor d)$
$\land (a \lor \neg c \lor \neg h)$
$\land (a \lor h \lor \neg m)$
$\land (b \lor k)$
$\land (\neg g \lor \neg c \lor i)$
$\land (\neg g \lor h \lor \neg i)$
$\land (g \lor h \lor \neg j)$
$\land (g \lor j \lor \neg m)$

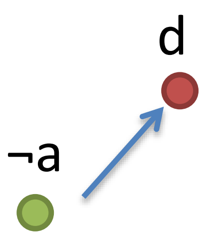




Learning

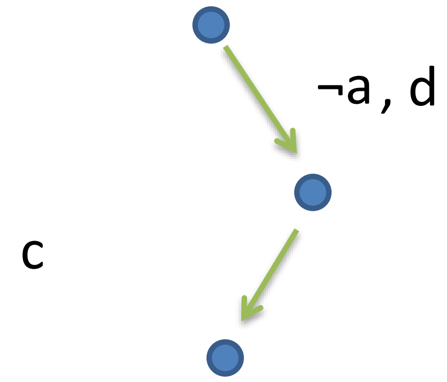
$$
\begin{aligned}
& (a \lor d) \\
\land\ & (a \lor \neg c \lor \neg h) \\
\land\ & (a \lor h \lor \neg m) \\
\land\ & (b \lor k) \\
\land\ & (\neg g \lor \neg c \lor i) \\
\land\ & (\neg g \lor h \lor \neg i) \\
\land\ & (g \lor h \lor \neg j) \\
\land\ & (g \lor j \lor \neg m)
\end{aligned}
$$

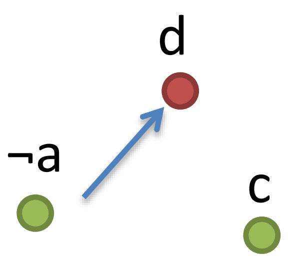




Learning

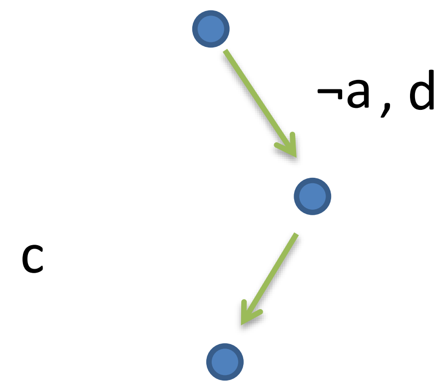
$$
\begin{aligned}
& (a \lor d) \\
\land\ & (a \lor \neg c \lor \neg h) \\
\land\ & (a \lor h \lor \neg m) \\
\land\ & (b \lor k) \\
\land\ & (\neg g \lor \neg c \lor i) \\
\land\ & (\neg g \lor h \lor \neg i) \\
\land\ & (g \lor h \lor \neg j) \\
\land\ & (g \lor j \lor \neg m)
\end{aligned}
$$

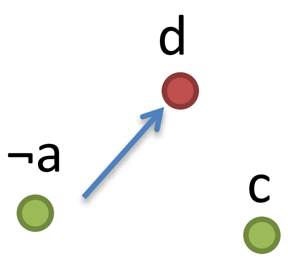




Learning

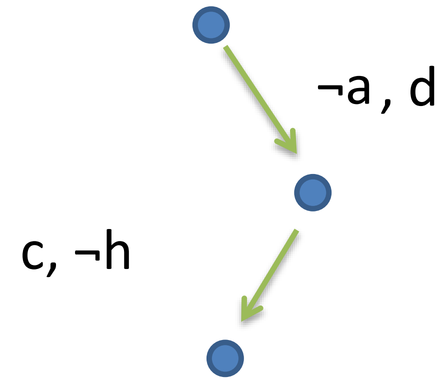
$$
\begin{aligned}
& (a \lor d) \\
\land\ & (a \lor \neg c \lor \neg h) \\
\land\ & (a \lor h \lor \neg m) \\
\land\ & (b \lor k) \\
\land\ & (\neg g \lor \neg c \lor i) \\
\land\ & (\neg g \lor h \lor \neg i) \\
\land\ & (g \lor h \lor \neg j) \\
\land\ & (g \lor j \lor \neg m)
\end{aligned}
$$

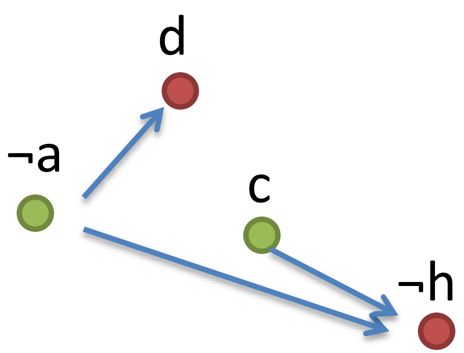




Learning

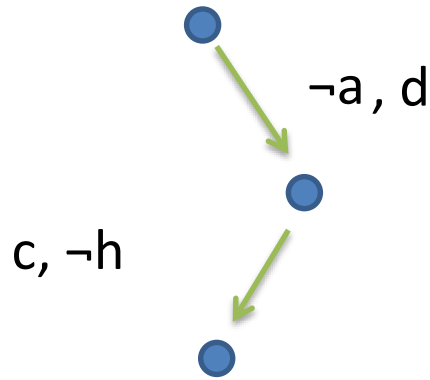
$$
\begin{aligned}
& (a \lor d) \\
\land\ & (a \lor \neg c \lor \neg h) \\
\land\ & (a \lor h \lor \neg m) \\
\land\ & (b \lor k) \\
\land\ & (\neg g \lor \neg c \lor i) \\
\land\ & (\neg g \lor h \lor \neg i) \\
\land\ & (g \lor h \lor \neg j) \\
\land\ & (g \lor j \lor \neg m)
\end{aligned}
$$

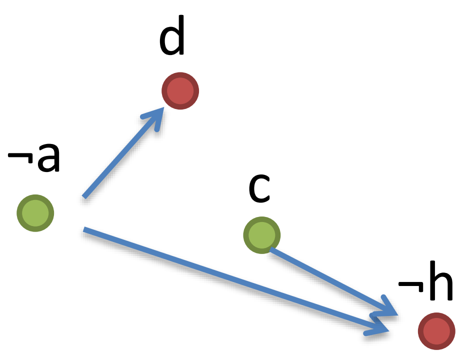




Learning

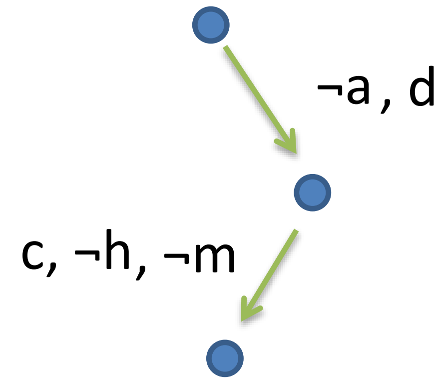
(a ∨ d)
∧ (a ∨ ¬c ∨ ¬h)
∧ (a ∨ h ∨ ¬m)
∧ (b ∨ k)
∧ (¬g ∨ ¬c ∨ i)
∧ (¬g ∨ h ∨ ¬i)
∧ (g ∨ h ∨ ¬j)
∧ (g ∨ j ∨ ¬m)

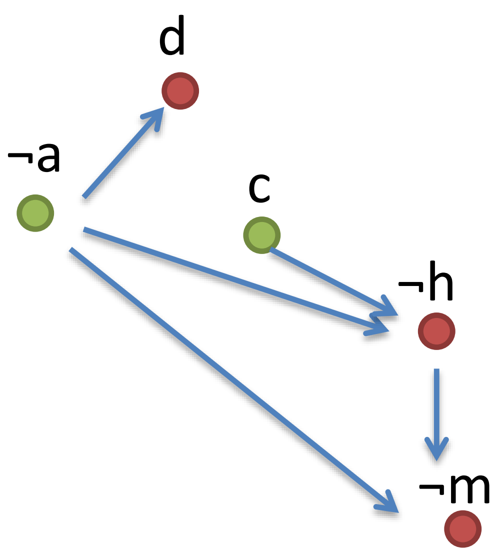




Learning

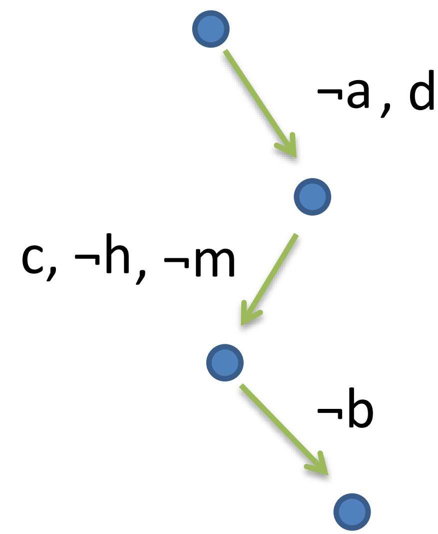
(a ∨ d)
∧ (a ∨ ¬c ∨ ¬h)
∧ (a ∨ h ∨ ¬m)
∧ (b ∨ k)
∧ (¬g ∨ ¬c ∨ i)
∧ (¬g ∨ h ∨ ¬i)
∧ (g ∨ h ∨ ¬j)
∧ (g ∨ j ∨ ¬m)

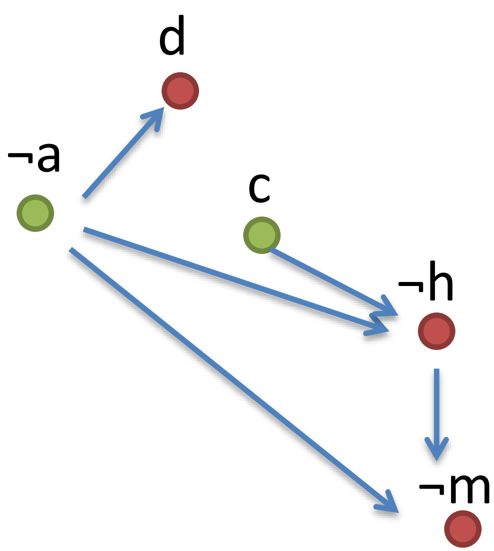




Learning

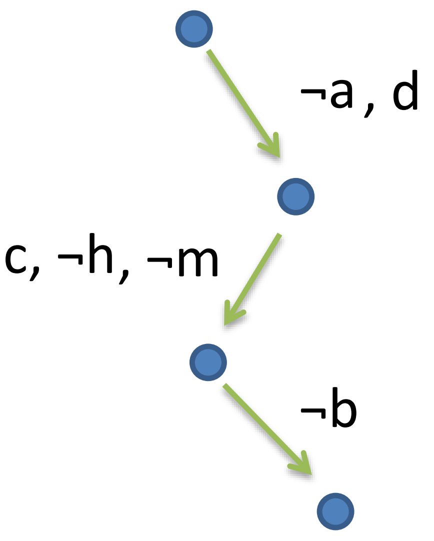
$$
\begin{aligned}
& (a \lor d) \\
\land\ & (a \lor \neg c \lor \neg h) \\
\land\ & (a \lor h \lor \neg m) \\
\land\ & (b \lor k) \\
\land\ & (\neg g \lor \neg c \lor i) \\
\land\ & (\neg g \lor h \lor \neg i) \\
\land\ & (g \lor h \lor \neg j) \\
\land\ & (g \lor j \lor \neg m)
\end{aligned}
$$

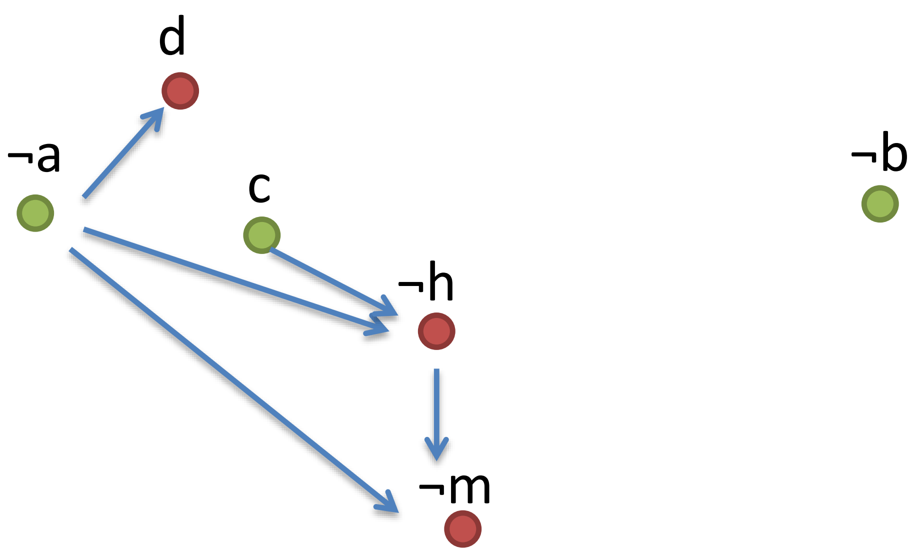




Learning

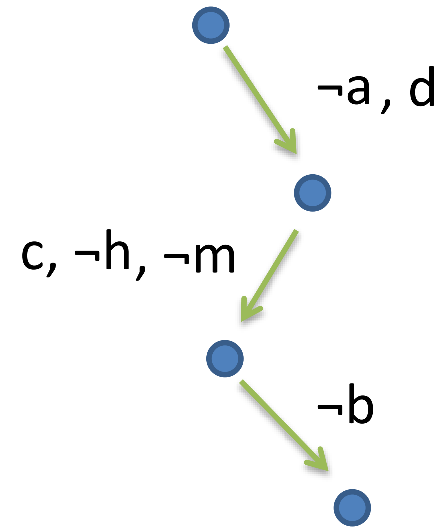
$(a \lor d)$
$\land\ (a \lor \neg c \lor \neg h)$
$\land\ (a \lor h \lor \neg m)$
$\land\ (b \lor k)$
$\land\ (\neg g \lor \neg c \lor i)$
$\land\ (\neg g \lor h \lor \neg i)$
$\land\ (g \lor h \lor \neg j)$
$\land\ (g \lor j \lor \neg m)$

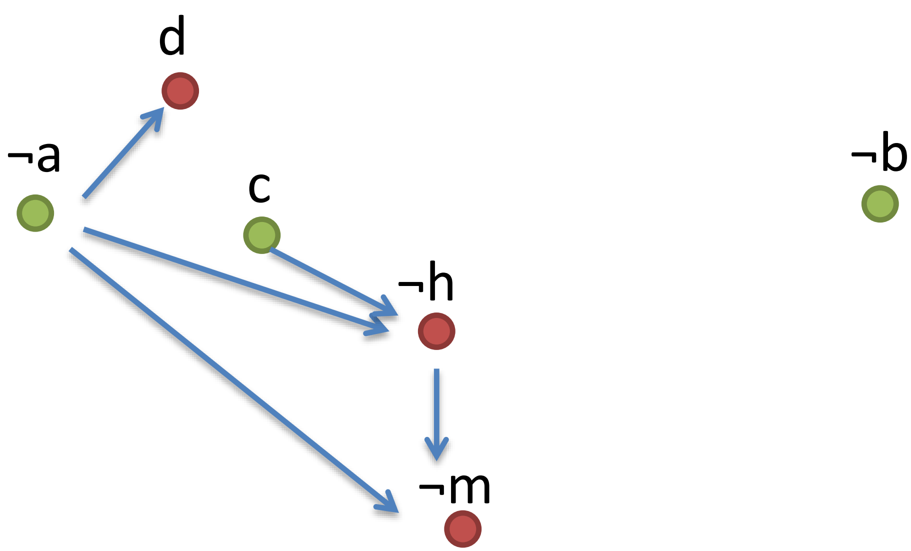




Learning

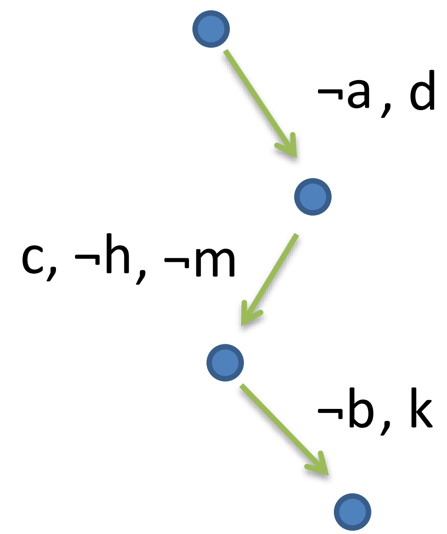
(a ∨ d)
∧ (a ∨ ¬c ∨ ¬h)
∧ (a ∨ h ∨ ¬m)
∧ (b ∨ k)
∧ (¬g ∨ ¬c ∨ i)
∧ (¬g ∨ h ∨ ¬i)
∧ (g ∨ h ∨ ¬j)
∧ (g ∨ j ∨ ¬m)

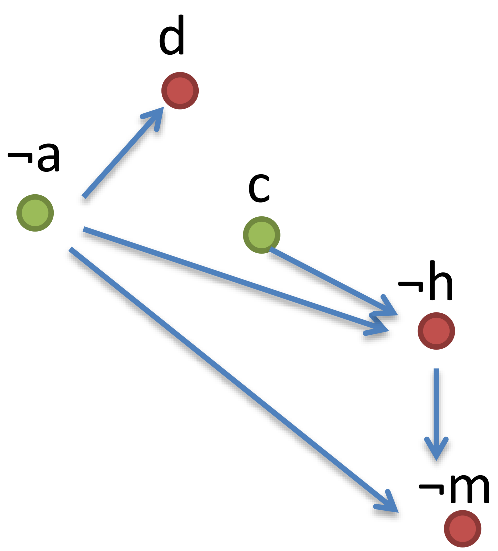


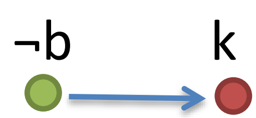


Learning

(a ∨ d)
∧ (a ∨ ¬c ∨ ¬h)
∧ (a ∨ h ∨ ¬m)
∧ (b ∨ k)
∧ (¬g ∨ ¬c ∨ i)
∧ (¬g ∨ h ∨ ¬i)
∧ (g ∨ h ∨ ¬j)
∧ (g ∨ j ∨ ¬m)

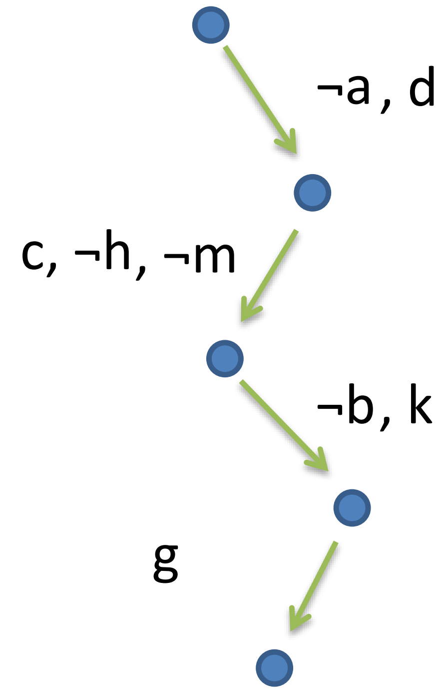


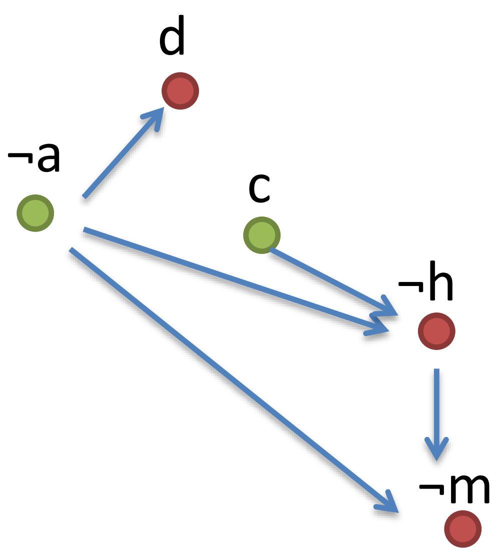


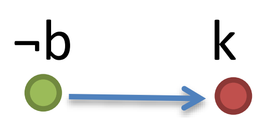


Learning

$$
\begin{aligned}
& (a \lor d) \\
\land\ & (a \lor \neg c \lor \neg h) \\
\land\ & (a \lor h \lor \neg m) \\
\land\ & (b \lor k) \\
\land\ & (\neg g \lor \neg c \lor i) \\
\land\ & (\neg g \lor h \lor \neg i) \\
\land\ & (g \lor h \lor \neg j) \\
\land\ & (g \lor j \lor \neg m)
\end{aligned}
$$

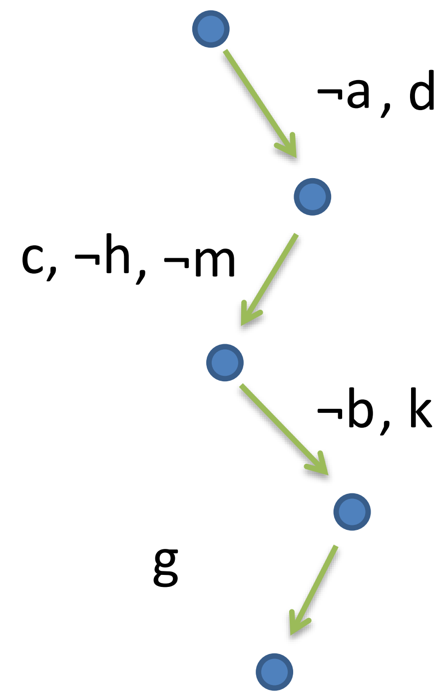


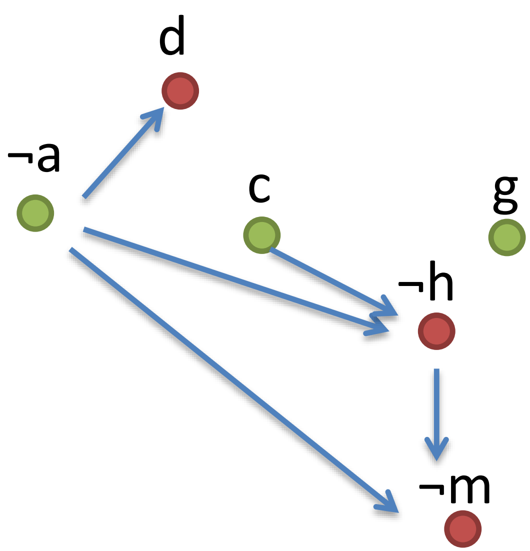


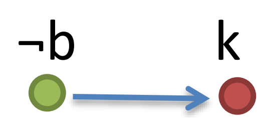


Learning

(a ∨ d)
∧ (a ∨ ¬c ∨ ¬h)
∧ (a ∨ h ∨ ¬m)
∧ (b ∨ k)
∧ (¬g ∨ ¬c ∨ i)
∧ (¬g ∨ h ∨ ¬i)
∧ (g ∨ h ∨ ¬j)
∧ (g ∨ j ∨ ¬m)

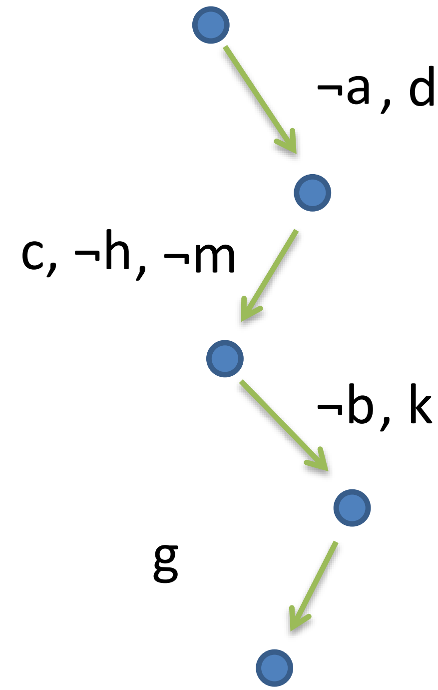


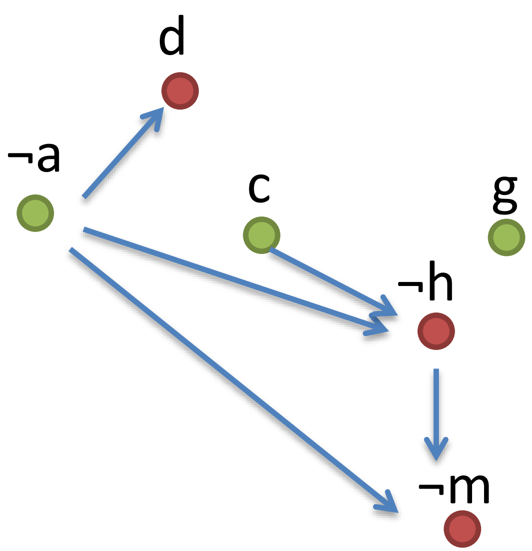


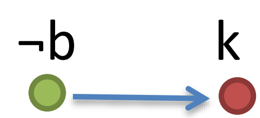


Learning

$$
\begin{aligned}
& (a \lor d) \\
\land\ & (a \lor \neg c \lor \neg h) \\
\land\ & (a \lor h \lor \neg m) \\
\land\ & (b \lor k) \\
\land\ & (\neg g \lor \neg c \lor i) \\
\land\ & (\neg g \lor h \lor \neg i) \\
\land\ & (g \lor h \lor \neg j) \\
\land\ & (g \lor j \lor \neg m)
\end{aligned}
$$

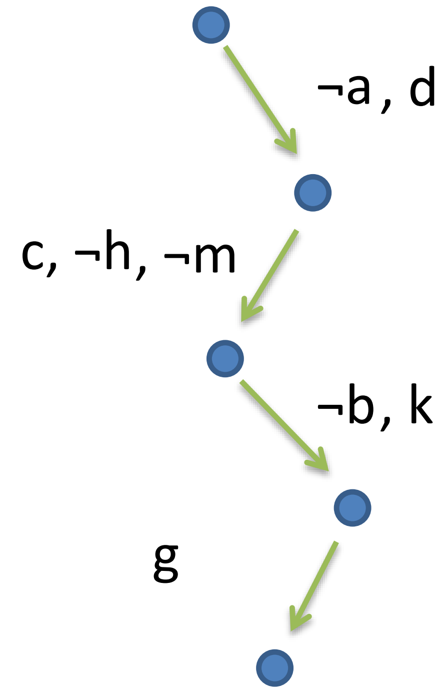


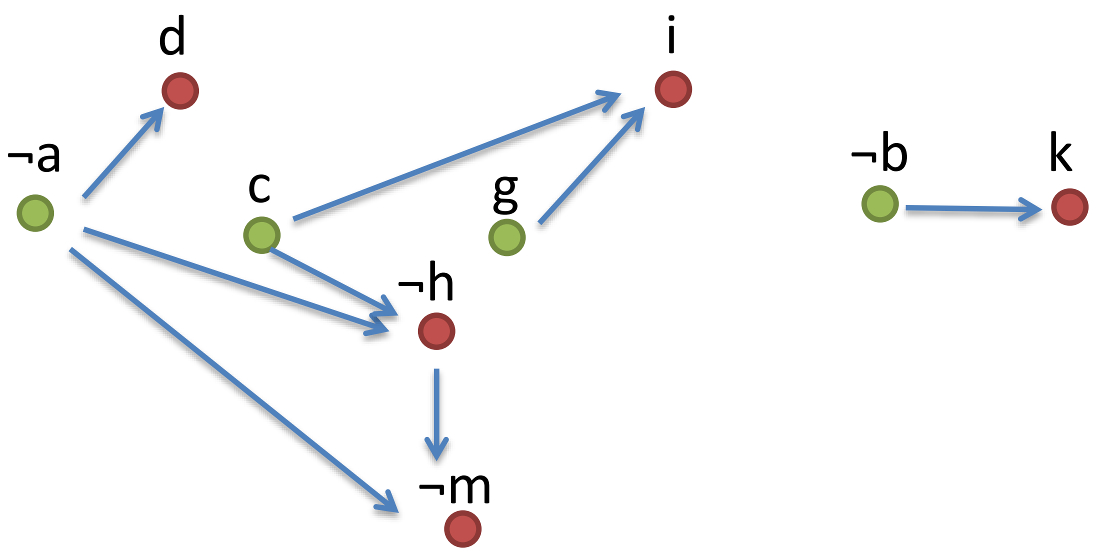




Learning

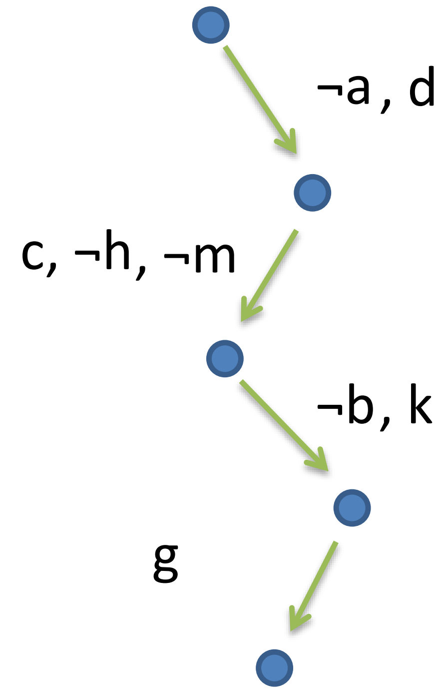
$$
\begin{aligned}
& (a \lor d) \\
\land\ & (a \lor \neg c \lor \neg h) \\
\land\ & (a \lor h \lor \neg m) \\
\land\ & (b \lor k) \\
\land\ & (\neg g \lor \neg c \lor i) \\
\land\ & (\neg g \lor h \lor \neg i) \\
\land\ & (g \lor h \lor \neg j) \\
\land\ & (g \lor j \lor \neg m)
\end{aligned}
$$

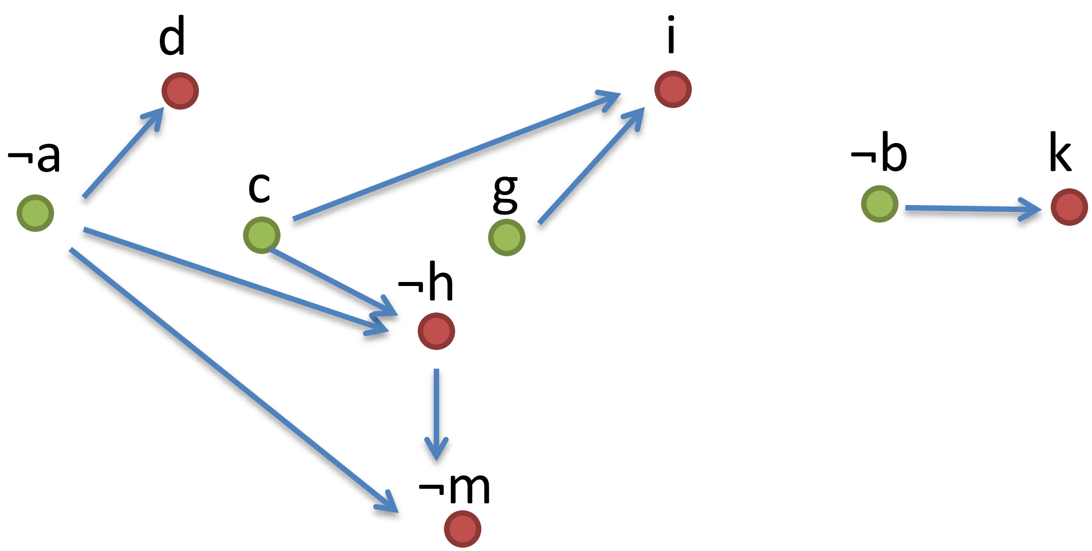




Learning

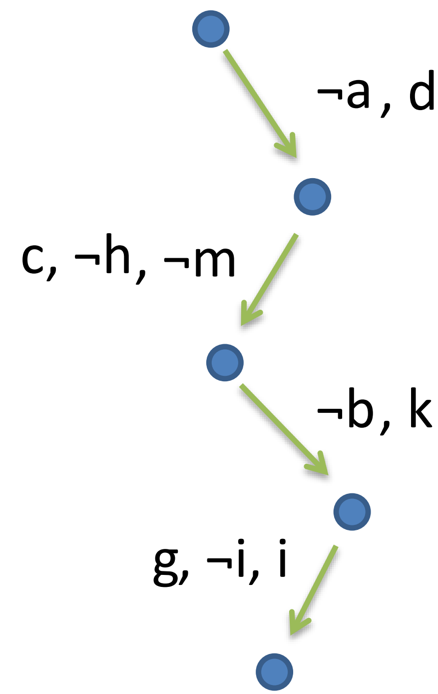
$$(a \lor d)$$
$$\land\ (a \lor \neg c \lor \neg h)$$
$$\land\ (a \lor h \lor \neg m)$$
$$\land\ (b \lor k)$$
$$\land\ (\neg g \lor \neg c \lor i)$$
$$\land\ (\neg g \lor h \lor \neg i)$$
$$\land\ (g \lor h \lor \neg j)$$
$$\land\ (g \lor j \lor \neg m)$$

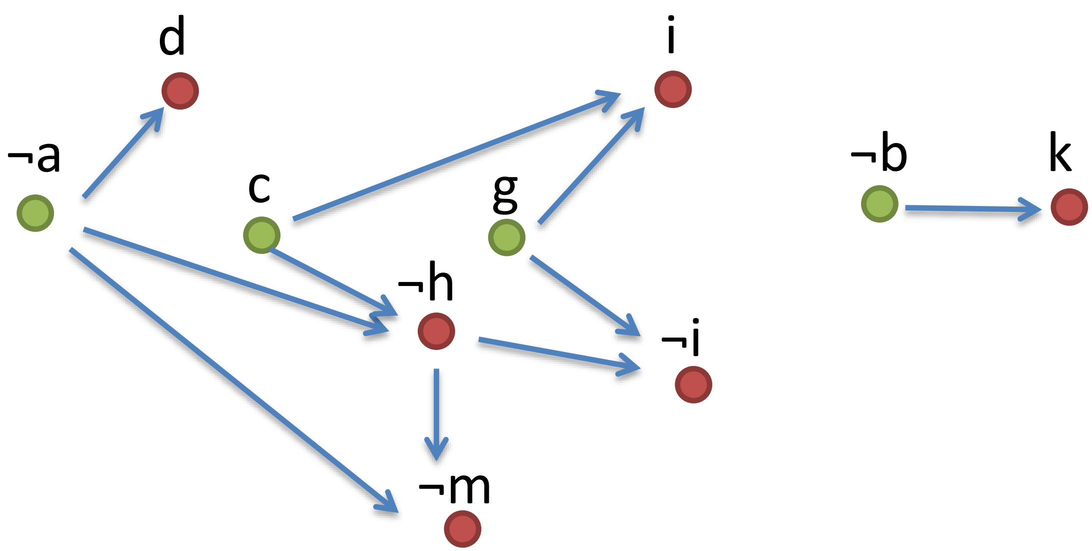




Learning

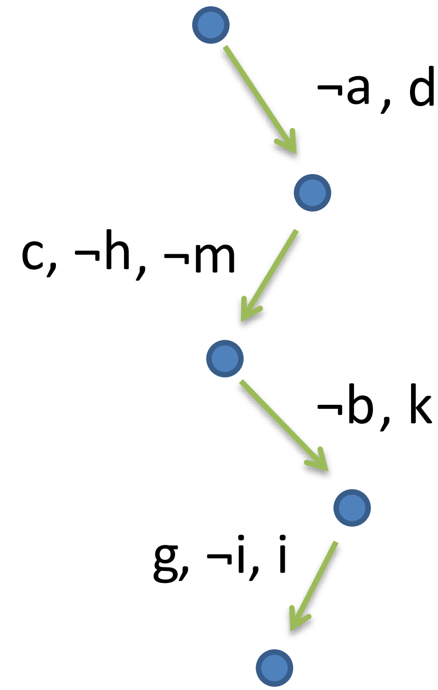
$$
\begin{array}{l}
\quad (a \lor d) \\
\land\ (a \lor \neg c \lor \neg h) \\
\land\ (a \lor h \lor \neg m) \\
\land\ (b \lor k) \\
\land\ (\neg g \lor \neg c \lor i) \\
\land\ (\neg g \lor h \lor \neg i) \\
\land\ (g \lor h \lor \neg j) \\
\land\ (g \lor j \lor \neg m)
\end{array}
$$

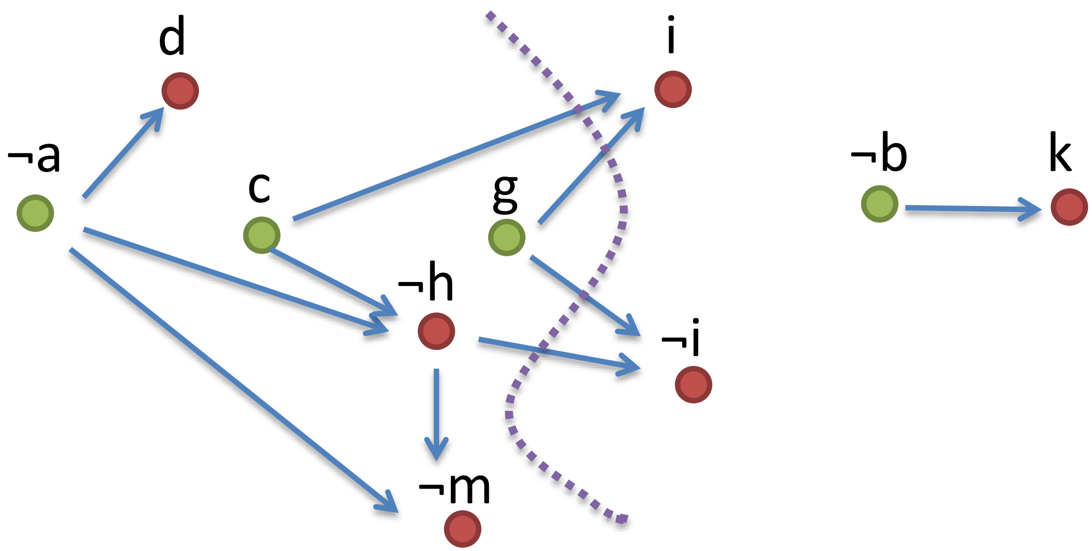




Learning

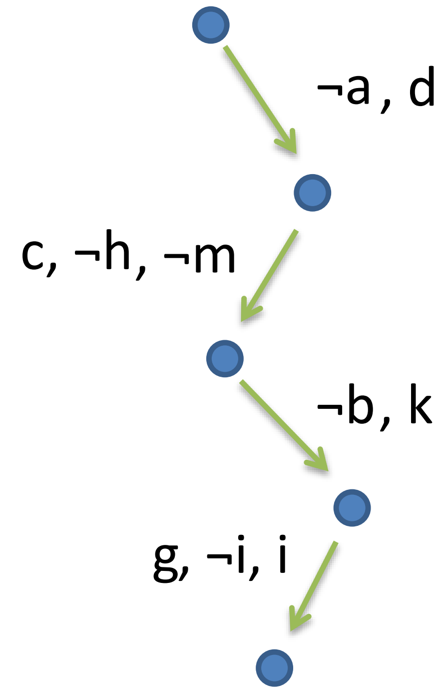
(a ∨ d)
∧ (a ∨ ¬c ∨ ¬h)
∧ (a ∨ h ∨ ¬m)
∧ (b ∨ k)
∧ (¬g ∨ ¬c ∨ i)
∧ (¬g ∨ h ∨ ¬i)
∧ (g ∨ h ∨ ¬j)
∧ (g ∨ j ∨ ¬m)

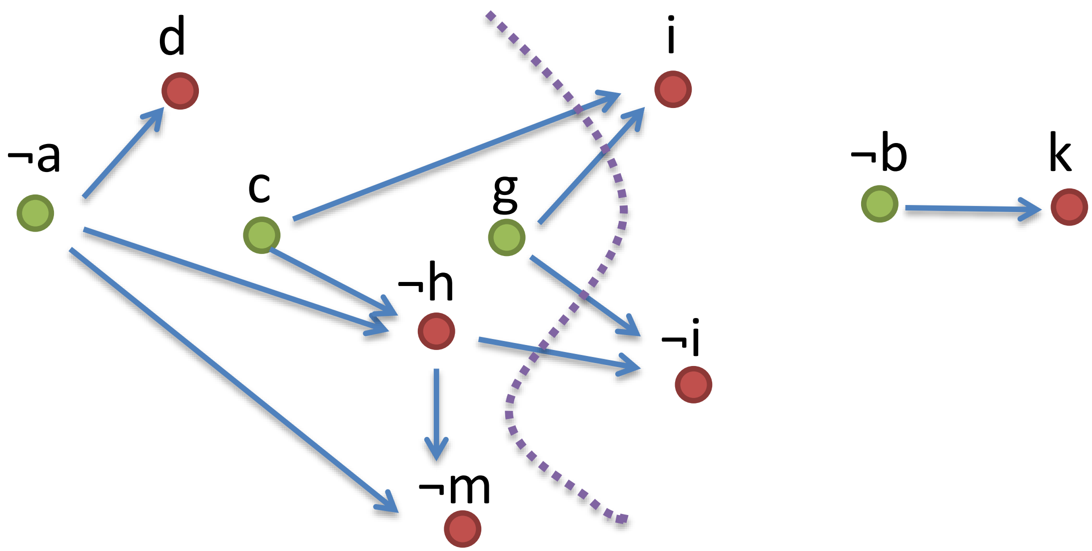




Learning

$$
\begin{aligned}
& (a \lor d) \\
\land\ & (a \lor \neg c \lor \neg h) \\
\land\ & (a \lor h \lor \neg m) \\
\land\ & (b \lor k) \\
\land\ & (\neg g \lor \neg c \lor i) \\
\land\ & (\neg g \lor h \lor \neg i) \\
\land\ & (g \lor h \lor \neg j) \\
\land\ & (g \lor j \lor \neg m)
\end{aligned}
$$

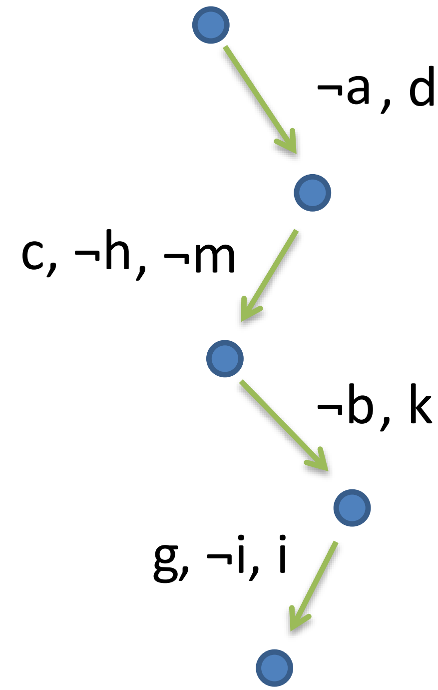


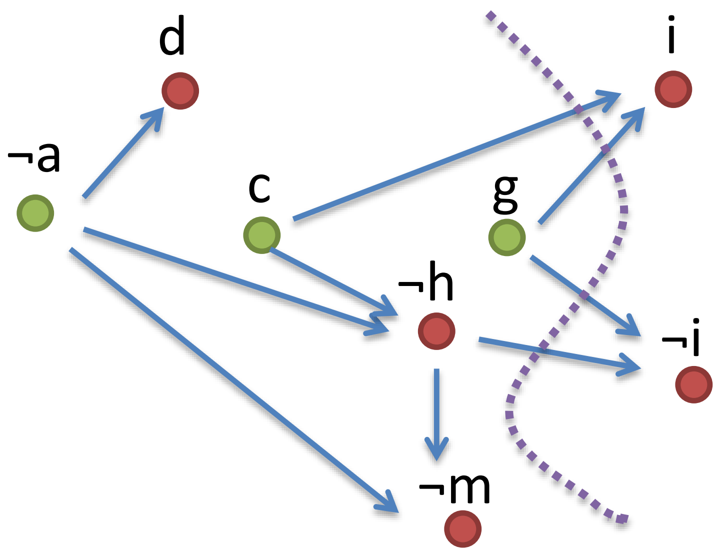


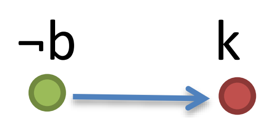
$\neg(c \land g \land \neg h)$

Learning

(a ∨ d)
∧ (a ∨ ¬c ∨ ¬h)
∧ (a ∨ h ∨ ¬m)
∧ (b ∨ k)
∧ (¬g ∨ ¬c ∨ i)
∧ (¬g ∨ h ∨ ¬i)
∧ (g ∨ h ∨ ¬j)
∧ (g ∨ j ∨ ¬m)







¬(c ∧ g ∧ ¬h)

…and backtrack to c, then assert ¬g !

Learning

- Learning has a dramatically positive impact.

- Learning also makes *restarts* possible:
  - Idea: after some number of literal assignments, drop the assignment stack and restart from zero.
  - Goal: avoid locally difficult subtrees.
  - Clauses encode previous knowledge and make new search faster.

Picking Variable Assignments

- Potential strategies:
  - Fixed ordering,
  - Frequency based,
  - “Maximal impact”.

Picking Variable Assignments

- Potential strategies:
  - Fixed ordering,
  - Frequency based,
  - “Maximal impact”.

- Overall favorite are activity-based heuristics:
  - Pick variables that you have seen a lot in conflicts.
  - Decay weights to favor recent conflicts.
  - Cheap to compute/update.

More Engineering…

- SAT dirty little secret: the enormous impact of preprocessing.
  - Problems are generated automatically (“compiled”); many redundancies, symmetry, etc.
  - Preprocessors look for subsumed clauses, equivalent clauses, etc.
  - Typically, run with timeout, then DPLL search.

More Engineering…

- SAT dirty little secret: the enormous impact of preprocessing.
  - Problems are generated automatically (“compiled”); many redundancies, symmetry, etc.
  - Preprocessors look for subsumed clauses, equivalent clauses, etc.
  - Typically, run with timeout, then DPLL search.

- Parallel SAT
  - State-of-the-art is to run instances with different parameters in parallel.

Scala Implementation

- CafeSat is a SAT solver written entirely in Scala
- It implements most of the techniques described in this lecture
- Some evaluation of the impact of various techniques on performance can be found in the paper:

  CafeSat: a modern SAT solver for Scala

https://github.com/regb/scabolic