SAT, CSP, and proofs

Ofer Strichman
Technion, Haifa

Tutorial HVC'13

The grand plan for today

- Intro: the role of SAT, CSP and proofs in verification
- SAT – how it works, and how it produces proofs
- CSP - how it works, and how it produces proofs
- Making proofs smaller

Why SAT ?

Example: is $(x_1 \wedge (x_2 \vee \neg x_1))$ satisfiable ?

$x_1, x_2 \in \mathcal{B}$

- Applications in verification:
  - Formal verification:
    - (Bounded) model checking for hardware [1999 --]
      - Over a dozen commerncial tools
    - (Bounded) model checking for software [2001 --]
      - CBMC, SAT-ABS, CLLVM, …
    - Satisfiability Modulo Theories (SMT) [2003 --]
      - e.g. MS – Z3 used in dozens of software analysis tools (SymDiff, VCC, Havoc, Spec#, …)

Why SAT ?

Example: is $(x_1 \wedge (x_2 \vee \neg x_1))$ satisfiable ?

$x_1, x_2 \in \mathcal{B}$

- Applications in verification:
  - Simulation:
    - Test generation for hardware
    - Test generation for software via SMT
      - MS-SAGE, KLEE, …

Why CSP (Constraints Satisfaction Problem) ?

Example:

is $(\mathrm{AllDiff}(x_1, x_2, x_3) \vee x_1 < x_2 + 3 \wedge x_2 > x_3 - 1))$ satisfiable ?

$x_1, x_2, x_3 \in [0..10] \cap \mathcal{Z}$

- Applications in verification:
  - Formal verification: ??
  - Simulation: test generation for hardware

Why CSP (Constraints Satisfaction Problem) ?

Example:

is (AllDiff(x_1 ,x_2, x_3) $\vee$ $x_1 < x_2 + 3 \wedge x_2 > x_3 - 1$)) satisfiable ?

$x_1, x_2, x_3 \in [0..10] \cap \mathcal{Z}$

- A Higher-level modeling language
  - Can lead to an order of magniture smaller model size.
    - Does not matter much in practice
- Certain constraints can be solved faster than in SAT
  - Some (e.g. “all-different”) can be solved directly in P

SAT and CSP

- SAT is crawling towards CSP
  - Various SAT solvers now support high-level constraints over Boolean variables:
    - Cripto-minisat supports XOR constraints
    - MiniSat+ supports cardinality constraints $\sum w_i x_i \leq c$
- CSP is crawling towards SAT:
  - Some solvers support reduction to SAT
  - Solution strategy now mimics SAT

Why proofs ?

- Traditionally the focus was on finding models
  - No information was given in case of UNSAT

- As of Chaff (2003 --) solvers produce proofs
  - Originally just to validate result

Why proofs ?

Several killer-applications (SAT):

- Validate UNSAT results
- From the proof we can extract an unsat core
  - Used in formal verification [AM03, KKB09, BKOSSB07…]
- Uses of the proof itself:
  - Interpolation-based model checking [M03].

- Can we foresee usage for proofs in CSP ?

The grand plan for today

- Intro: the role of SAT, CSP and proofs in verification
- SAT – how it works, and how it produces proofs
- CSP - how it works, and how it produces proofs
- Making proofs smaller

CNF-SAT

- Conjunctive Normal Form: Conjunction of disjunction of literals. Example:

$$(\neg x_1 \vee \neg x_2) \wedge (x_2 \vee x_4 \vee \neg x_1) \wedge \ldots$$

- Polynomial transformation to CNF due to Tseitin (1970)
  - Requires adding auxiliary variables.

Main steps – SAT

SAT

- "Decide"
  - Variable, value

- "Boolean Constraints Propagation (BCP)"
  - infer implied assignments

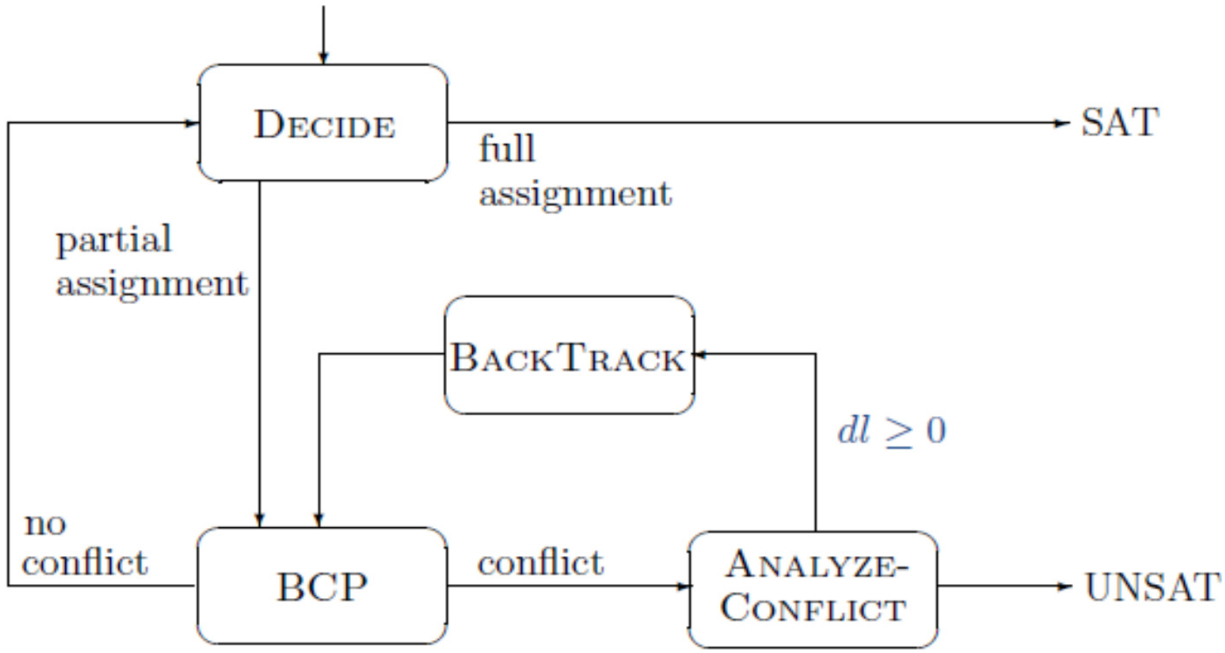
- "Analyze conflict"
  - applies learning
  - computes backtracking level

About that "constraints propagation"

- given $(\neg x_1 \vee \neg x_2) \wedge (x_2 \vee x_4 \vee \neg x_1) \wedge ...$

  BCP: $x_1 = 1 \Rightarrow x_2 = 0 \Rightarrow x_4 = 1 \Rightarrow \ldots$

SAT essentials



Implication graphs and learning

Current truth assignment: $\{x_9=0@1, x_{10}=0@3, x_{11}=0@3, x_{12}=1@2, x_{13}=1@2\}$

Current decision assignment: $\{x_1=1@6\}$

$\omega_1 = (\neg x_1 \vee x_2)$

$\omega_2 = (\neg x_1 \vee x_3 \vee x_9)$

$\omega_3 = (\neg x_2 \vee \neg x_3 \vee x_4)$

$\omega_4 = (\neg x_4 \vee x_5 \vee x_{10})$

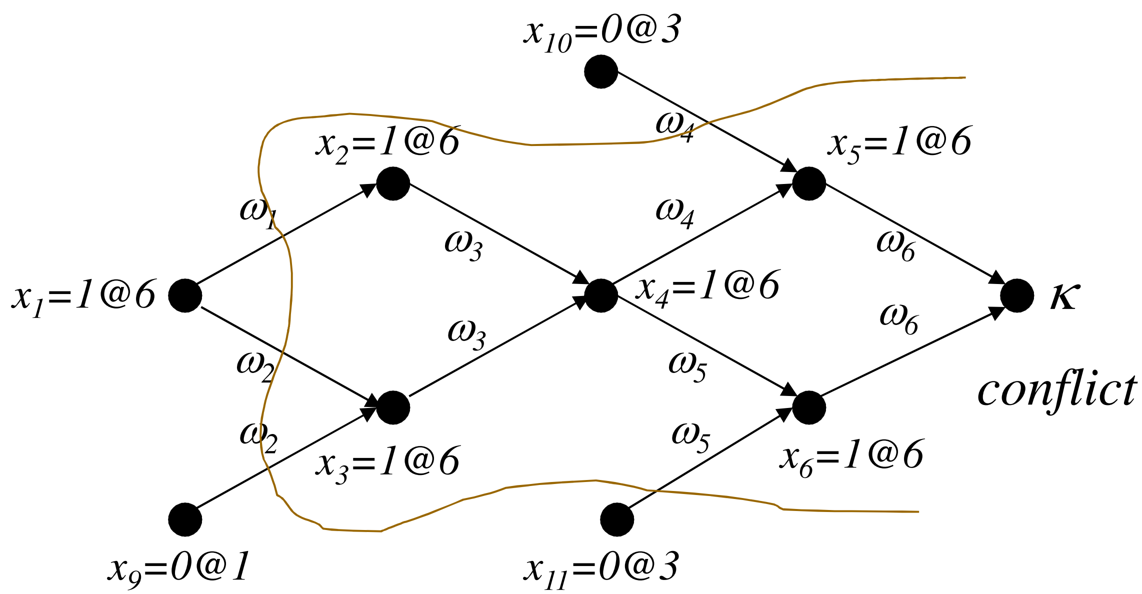
$\omega_5 = (\neg x_4 \vee x_6 \vee x_{11})$

$\omega_6 = (\neg x_5 \vee \neg x_6)$

$\omega_7 = (x_1 \vee x_7 \vee \neg x_{12})$

$\omega_8 = (x_1 \vee x_8)$

$\omega_9 = (\neg x_7 \vee \neg x_8 \vee \neg x_{13})$



We learn the *conflict clause* $\omega_{10} : (\neg x_1 \vee x_9 \vee x_{11} \vee x_{10})$

and backtrack to the highest (deepest) dec. level in this clause (6).

Implication graph, flipped assignment

$\omega_1 = (\neg x_1 \lor x_2)$

$\omega_2 = (\neg x_1 \lor x_3 \lor x_9)$

$\omega_3 = (\neg x_2 \lor \neg x_3 \lor x_4)$

$\omega_4 = (\neg x_4 \lor x_5 \lor x_{10})$

$\omega_5 = (\neg x_4 \lor x_6 \lor x_{11})$

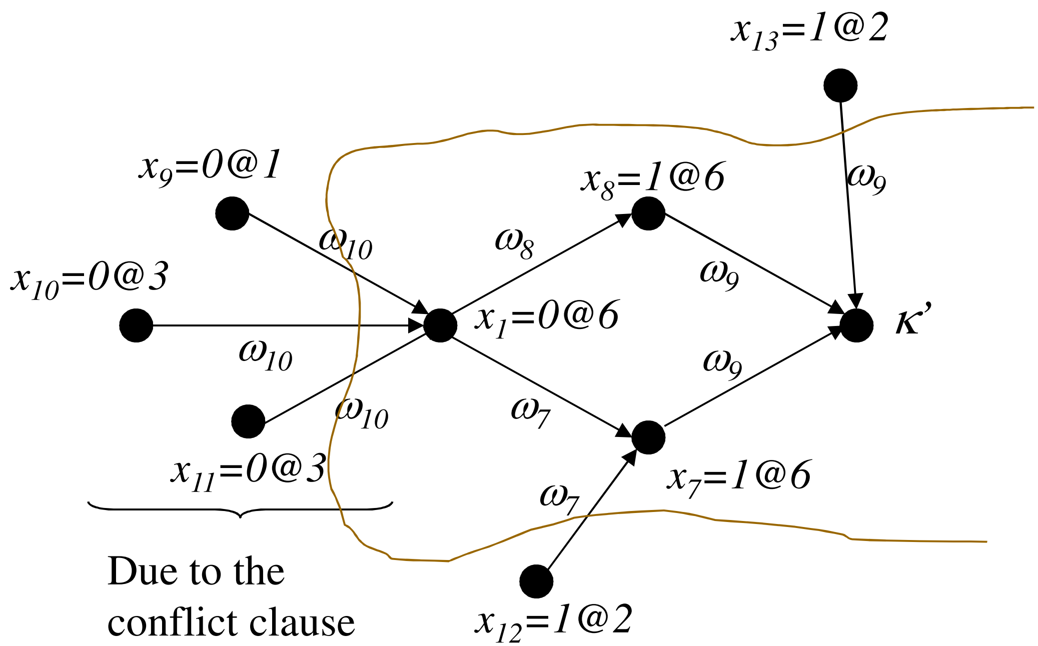
$\omega_6 = (\neg x_5 \lor x_6)$

$\omega_7 = (x_1 \lor x_7 \lor \neg x_{12})$

$\omega_8 = (x_1 \lor x_8)$

$\omega_9 = (\neg x_7 \lor \neg x_8 \lor \neg x_{13})$

$\omega_{10} : (\neg x_1 \lor x_9 \lor x_{11} \lor x_{10})$



We learn the *conflict clause* $\omega_{11} : (\neg x_{13} \lor x_9 \lor x_{10} \lor x_{11} \lor \neg x_{12})$

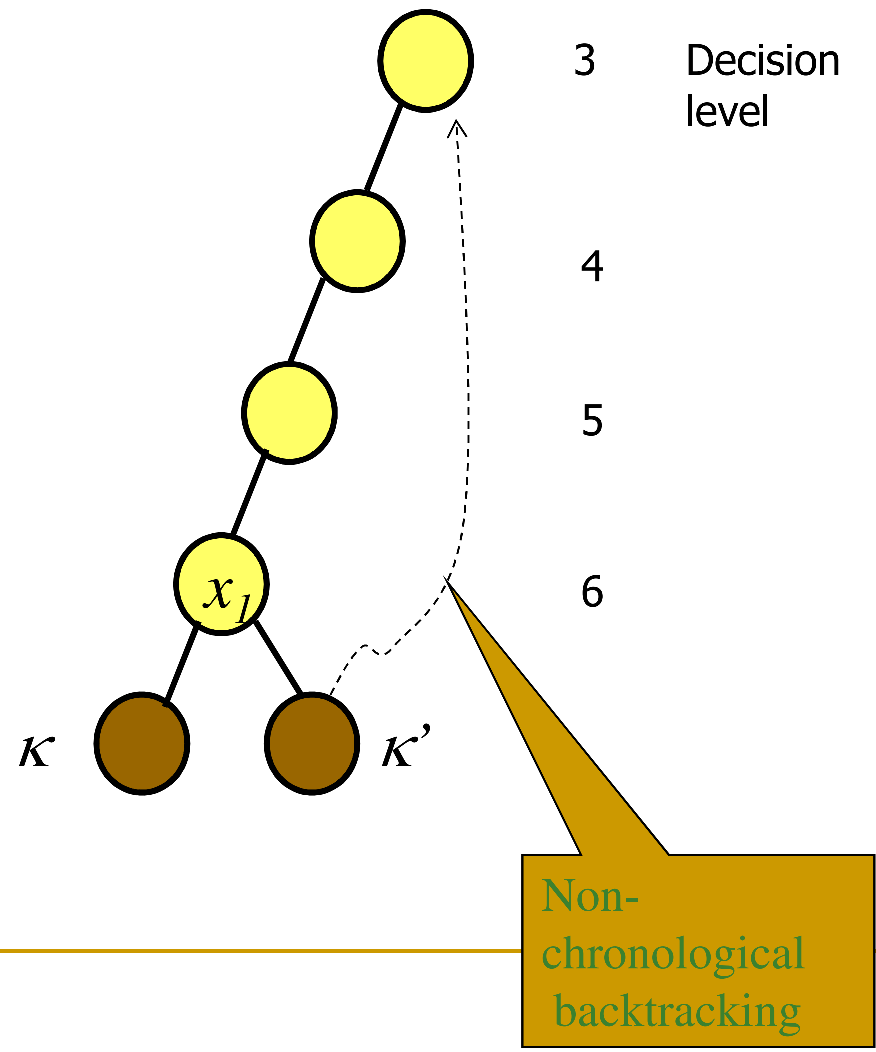
and backtrack to the highest (deepest) dec. level in this clause (3).

Non-chronological backtracking

Which assignments caused the conflicts ?

$x_9 = 0@1$
$x_{10} = 0@3$
$x_{11} = 0@3$
$x_{12} = 1@2$
$x_{13} = 1@2$

These assignments Are sufficient for Causing a conflict.



Backtrack to decision level 3

Learning and resolution

- Learning of a clause = inference by resolution.
  - To be explained

- This is the key for producing a machine-checkable proof

Resolution

- …By example:

$$\frac{(x_1 \vee x_2) \qquad (\neg x_1 \vee x_3 \vee x_4)}{(x_2 \vee x_3 \vee x_4)}$$

- Formally:

$$\frac{(a_1 \vee \ldots \vee a_n \vee \beta) \qquad (b_1 \vee \ldots \vee b_m \vee (\neg \beta))}{(a_1 \vee \ldots \vee a_n \vee b_1 \vee \ldots \vee b_m)} \quad \text{(Binary Resolution)}$$

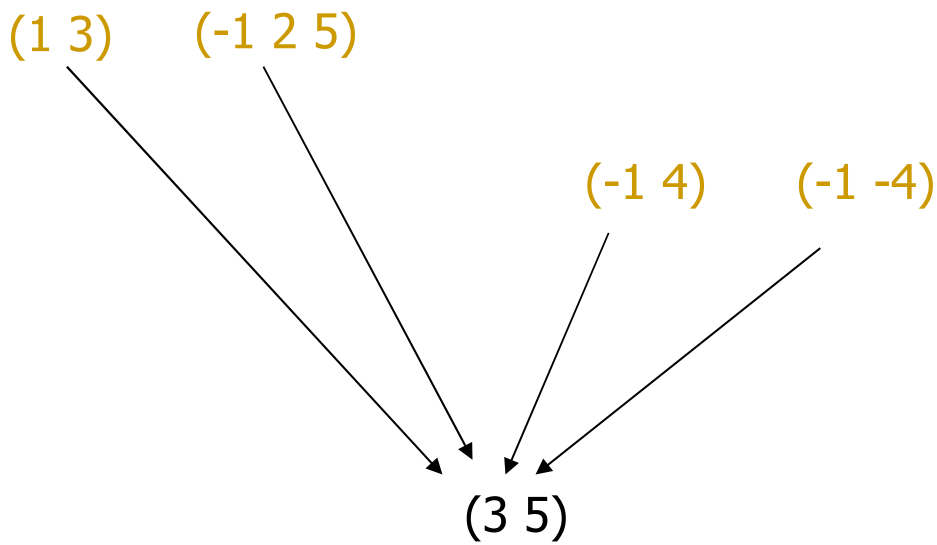
Resolution proof

A proof: $(1\ 3) \wedge (-1\ 2\ 5) \wedge (-1\ 4) \wedge (-1\ -4) \ \vdash\ (3\ 5)$

$$\begin{tikzcd}
(1\ 3) \arrow[dr] & & (-1\ 2\ 5) \arrow[dl] & & & & & \\
& (2\ 3\ 5) \arrow[dr] & & (1\ -2) \arrow[dl] & & (-1\ 4) \arrow[dr] & & (-1\ -4) \arrow[dl] \\
& & (1\ 3\ 5) \arrow[drr] & & & & (-1) \arrow[dll] & \\
& & & & (3\ 5) & & &
\end{tikzcd}$$

Resolution proof ⇒ Hyper resolution proof

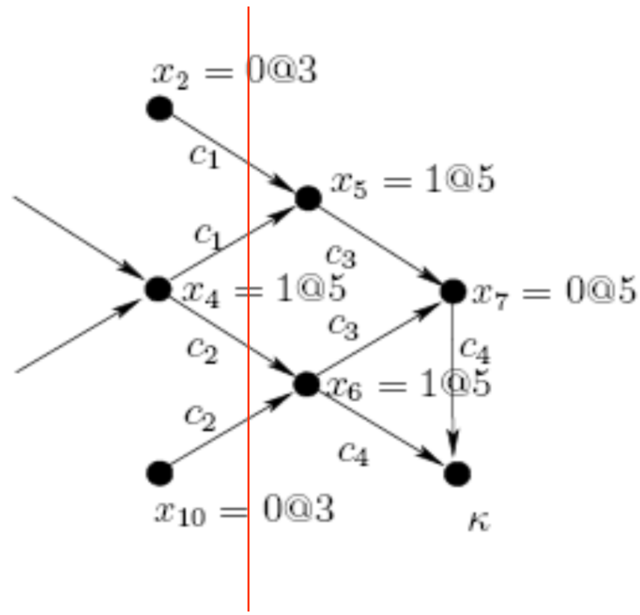
A proof: (1 3) ∧ (-1 2 5) ∧ (-1 4) ∧ (-1 -4) ⊢ (3 5)



Conflict clauses and resolution

- Consider the following example:

$$
\begin{aligned}
c_1 &= (\neg x_4 \vee x_2 \vee x_5) \\
c_2 &= (\neg x_4 \vee x_{10} \vee x_6) \\
c_3 &= (\neg x_5 \vee \neg x_6 \vee \neg x_7) \\
c_4 &= (\neg x_6 \vee x_7) \\
&\vdots \qquad \vdots
\end{aligned}
$$

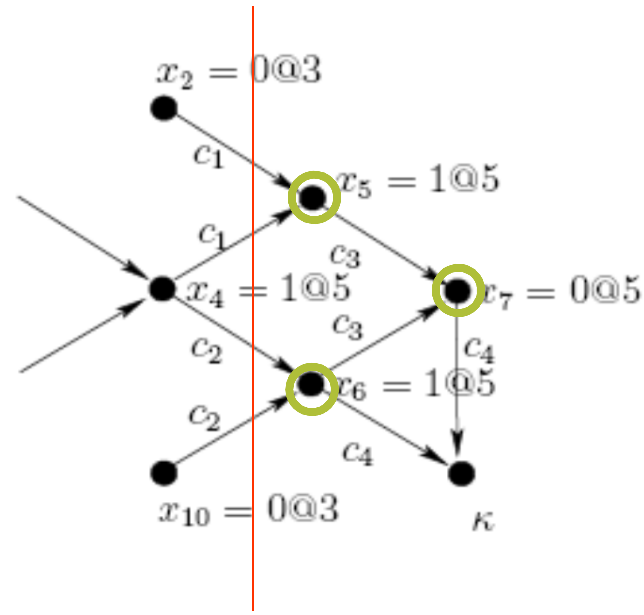


- Conflict clause: c_5: $(x_2 \vee \neg x_4 \vee x_{10})$
- We show that c_5 is inferred by resolution from $c_1, \ldots, c_4$

Conflict clauses and resolution

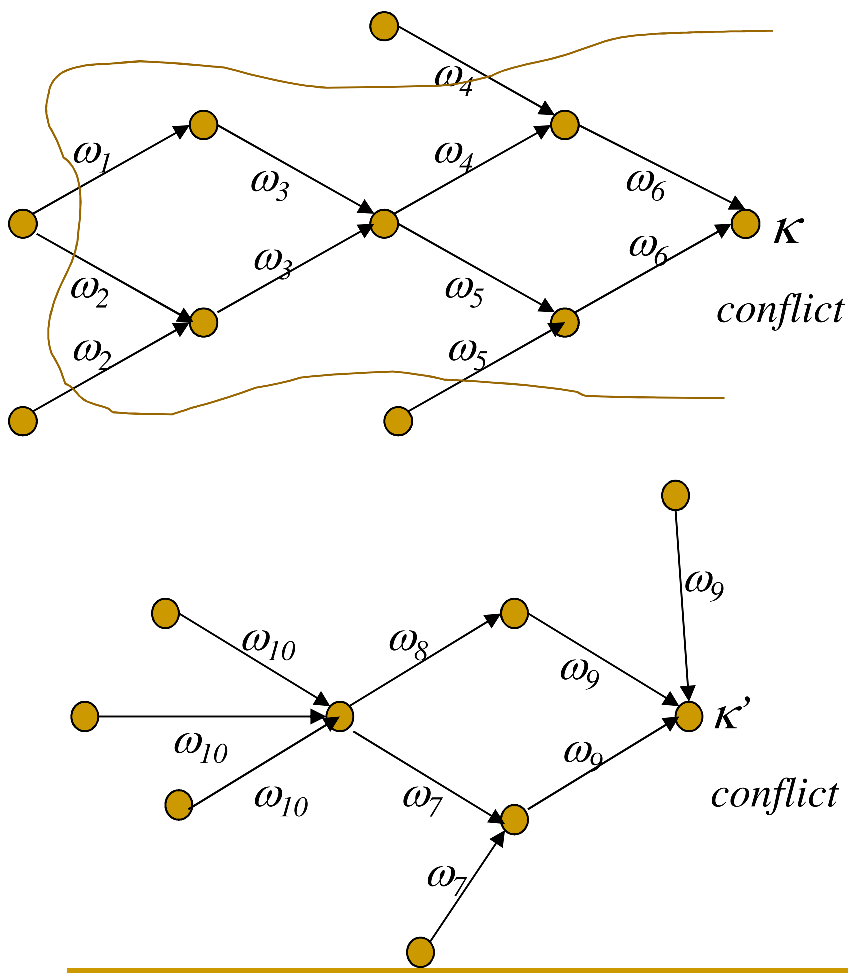
- Conflict clause: c_5: $(x_2 \vee \neg x_4 \vee x_{10})$

$$
\begin{aligned}
c_1 &= (\neg x_4 \vee x_2 \vee x_5) \\
c_2 &= (\neg x_4 \vee x_{10} \vee x_6) \\
c_3 &= (\neg x_5 \vee \neg x_6 \vee \neg x_7) \\
c_4 &= (\neg x_6 \vee x_7) \\
&\vdots \qquad \vdots
\end{aligned}
$$

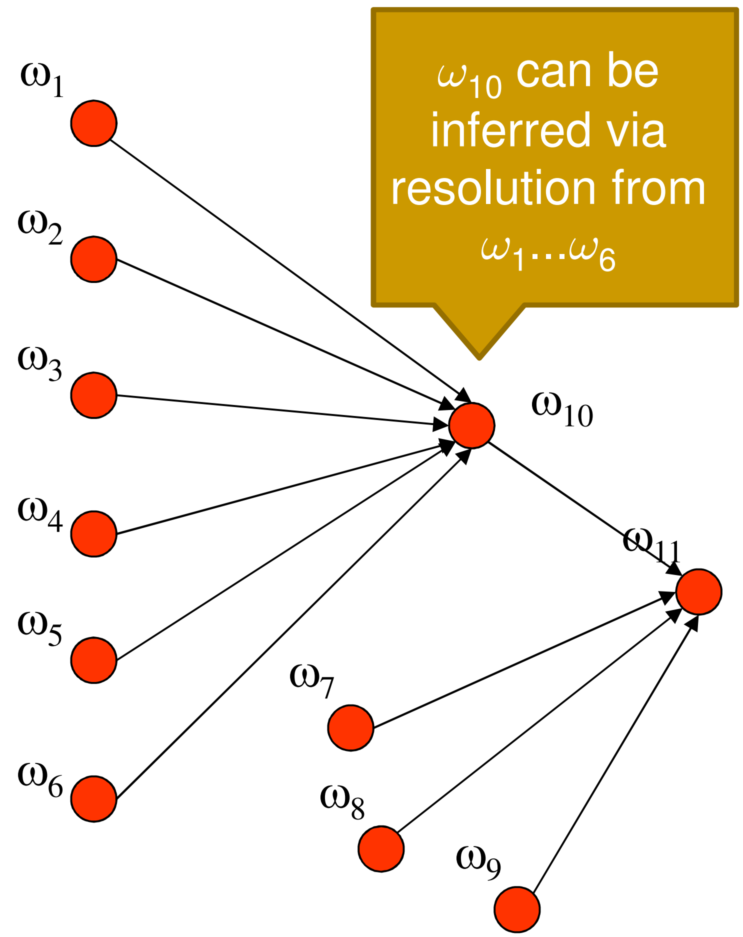


- BCP order: x_4, x_5, x_6, x_7
  - T1 = Res(c_4,c_3,x_7) = $(\neg x_5 \vee \neg x_6)$
  - T2 = Res(T1, c_2, x_6) = $(\neg x_4 \vee \neg x_5 \vee X_{10})$
  - T3 = Res(T2,c_1,x_5) = $\mathbf{(x_2 \vee \neg x_4 \vee x_{10})}$

The Resolution-Graph



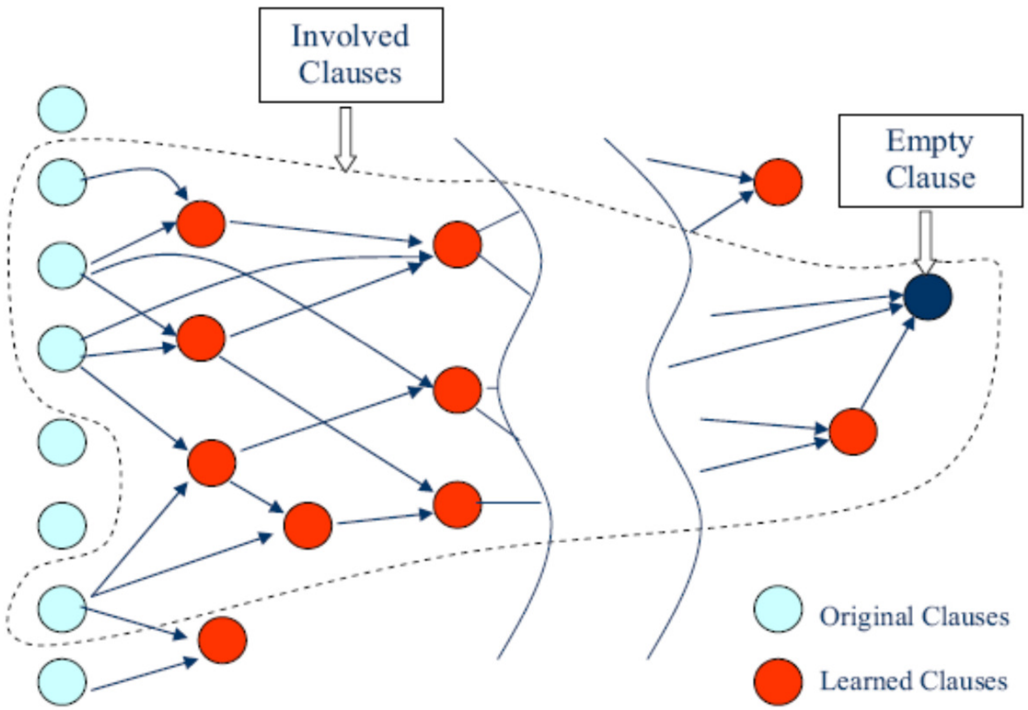
(Hyper) Resolution Graph



The resolution graph

What is it good for ?

Example: for computing an Unsatisfiable core



[Picture Borrowed from Zhang, Malik SAT'03]

The grand plan for today

- Intro: the role of SAT, CSP and proofs in verification
- SAT – how it works, and how it produces proofs
- <u>CSP - how it works, and how it produces proofs</u>
- Making proofs smaller

Main steps – SAT and CSP*

SAT

- "Decide"
  - Variable, value
- "Boolean Constraints Propagation (BCP)"
  - infer implied assignments
- "Analyze conflict"
  - applies learning
  - computes backtracking level

CSP

- *Same*
- "~~Boolean~~ Constraints Propagation (CP)"
  - same
- *Same*

*As implemented in PCS / Michael Veksler

About that "constraints propagation"

- Given $x_1, x_2, x_3 \in [1..3]$, AllDifferent(x_1, x_2, x_3)

  CP: $x_1 = 1 \Rightarrow x_2, x_3 \in [2..3]$

What about CSP proofs ?

- SAT solvers generate proofs:
  - From initial clauses to ( ).
  - Inference is via the binary-resolution rule.

- Unlike SAT solvers, CSPs:
  - have non-Boolean domains, and
  - non-clausal constraints.

- Can this gap be bridged?
  - The following is based on [SV10]

Signed CNF

... by examples:

- A positive signed literal: $a \in \{1, 2, 3\}$.
- A negative signed literal: $a \in \overline{\{1, 2, 4\}}$.
- A signed clause is a disjunction of signed literals. e.g.,

$$(a \in \{1, 5\} \vee b \in \overline{\{4\}})$$

Signed resolution

- A binary-resolution rule for signed-CNF:

$$\frac{(Literals_1 \vee x \in A) \qquad (x \in B \vee Literals_2)}{(Literals_1 \vee x \in A \cap B \vee Literals_2)} (\mathrm{sRes}(x))$$

- Signed-clauses ✓

- What about other constraints ?
  e.g. $\neq, \leq, \mathrm{allDifferent}(v_1, \ldots, v_k)$

  should we just convert CSP to signed CNF?

Signed resolution

- Q: should we just convert CSP to signed CNF?

- A: No, because it is generally inefficient:
  - e.g., x ≠ y requires:

$$(x \in \overline{\{1\}} \vee y \in \overline{\{1\}}) \wedge (x \in \overline{\{2\}} \vee y \in \overline{\{2\}}) \wedge \ldots$$

Towards a solution…

- Solution: introduce clauses *lazily*.

- Consider a general constraint *c*, such that:
  - In the context of $l_1 \wedge l_2 \wedge \ldots \wedge l_n$,
  - propagation of *c* implies l :

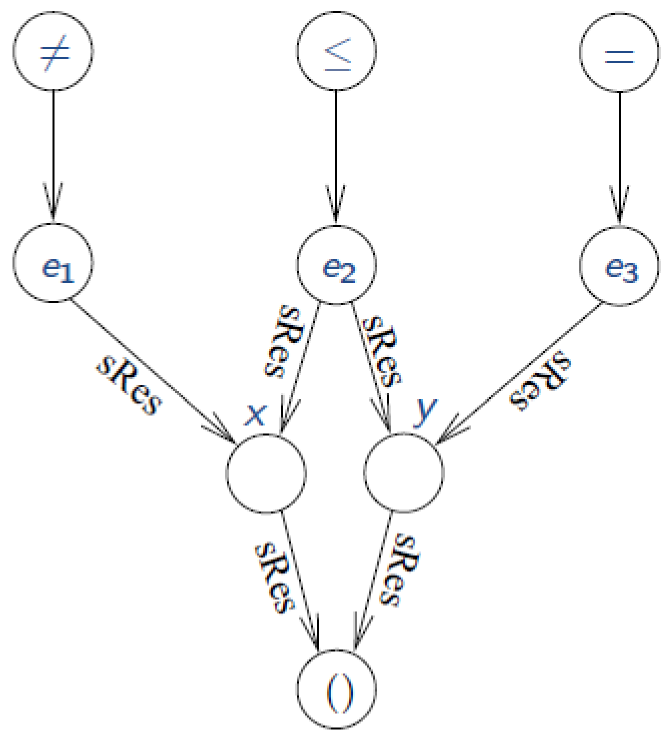
$$(l_1 \wedge l_2 \wedge \ldots \wedge l_n \wedge c) \rightarrow l$$

Towards a solution…

$$(I_1 \wedge I_2 \wedge \ldots \wedge I_n \wedge c) \rightarrow I$$

- Find an explanation clause e such that:
  - e is not too strong: $c \rightarrow e$
  - e is strong enough: $(I_1 \wedge I_2 \wedge \ldots \wedge I_n \wedge e) \rightarrow I$

The structure of a CSP proof



e_1, e_2, e_3 – explanation clauses.

Explanation rules

For every constraint there is an explanation clause:

$$\frac{\langle constraint \rangle}{\langle explanation\ clause \rangle}\ (\langle rule\ name \rangle)$$

Explanation rules – example 1

- Constraint: $x \neq y$

parameterized

$$\frac{x \neq y}{x \in \overline{\{m\}} \vee y \in \overline{\{m\}}} \, (Ne(m))$$

m = the value that trigerred the rule

Explanation rules – example 1

Propagation:

- context: $l_1 : \ (x = 1), \quad l_2 : \ (y \in [1..100])$.
- constraint: $c : \ x \neq y$.
- implies: $l : \ (y \in [2..100])$.

$$e : \quad (x \in \overline{\{1\}} \vee y \in \overline{\{1\}}) \qquad // \ Ne(1)$$

... *indeed*:

- $c \xrightarrow{Ne(1)} e$
- $(l_1 \wedge l_2 \wedge e) \longrightarrow l$

Explanation rules – example 2

- Constraint: $x \leq y$

$$\frac{x \leq y}{(x \in (-\infty, m] \vee y \in [m+1, \infty))} \, (LE(m))$$

Instantiate m with $max(domain(y))$

Explanation rules – example 2

Propagation:

- context: $l_1 : (x \in [1..3]),\ l_2 : (y \in [0..2])$
- constraint: $c : x \leq y$.
- implies: $l : x \in [1..2]$

Explanation:

- $e : (x \in (-\infty, 2] \vee y \in [3, \infty)).\quad // = LE(2)$

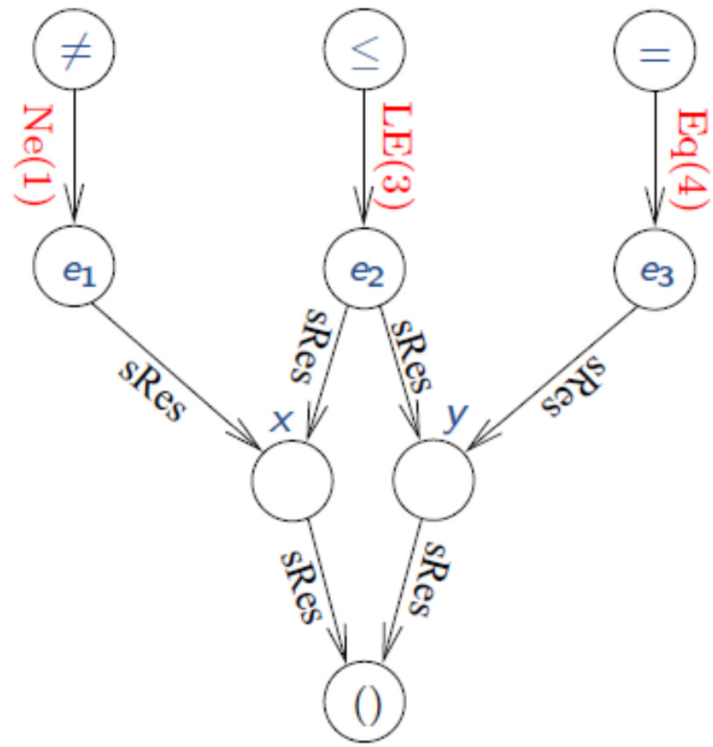
...*indeed*:

- $c \xrightarrow{\text{LE(2)}} e$
- $(l_1 \wedge l_2 \wedge e) \longrightarrow l$

Each constraint has its rule …

Constraint	Name	Inference rule
$a \neq b$	$\mathrm{Ne}(m)$	$\dfrac{a \neq b}{(a \neq m \vee b \neq m)}$
$x \leq y$	$\mathrm{LE}(m)$	$\dfrac{x \leq y}{(x \in (-\infty, m] \vee y \in [m+1, \infty))}$
$a = b$	$\mathrm{Eq}(D)$	$\dfrac{a = b}{(a \notin D \vee b \in D)}$
$a \leq b + c$	$LE_{+}(m, n)$	$\dfrac{a \leq b + c}{(a \in (-\infty, m+n] \vee b \in [m+1, \infty) \vee c \in [n+1, \infty))}$
$a = b + c$	$EQ^{a}_{+}(l_b, u_b, l_c, u_c)$	$\dfrac{a = b + c}{(a \in [l_b + l_c, u_b + u_c] \vee b \notin [l_b, u_b] \vee c \notin [l_c, u_c])}$
$\mathrm{AllDiff}(v_1, \ldots, v_k)$	$AD(D, V)$	$\dfrac{\mathrm{AllDiff}(v_1, \ldots, v_k)}{(\bigvee_{v \in V} v \notin D)}$
$\vdots$	$\vdots$	$\vdots$

So this is how the proof looks like…



e_1, e_2, e_3 – explanation clauses.

The grand plan for today

- Intro: the role of SAT, CSP and proofs in verification
- SAT – how it works, and how it produces proofs
- CSP - how it works, and how it produces proofs
- Making proofs smaller

Minimizing the core

- The proof is not unique.
  - Different proofs / different cores.

- Can we find a minimum / minimal / smaller cores/proofs?

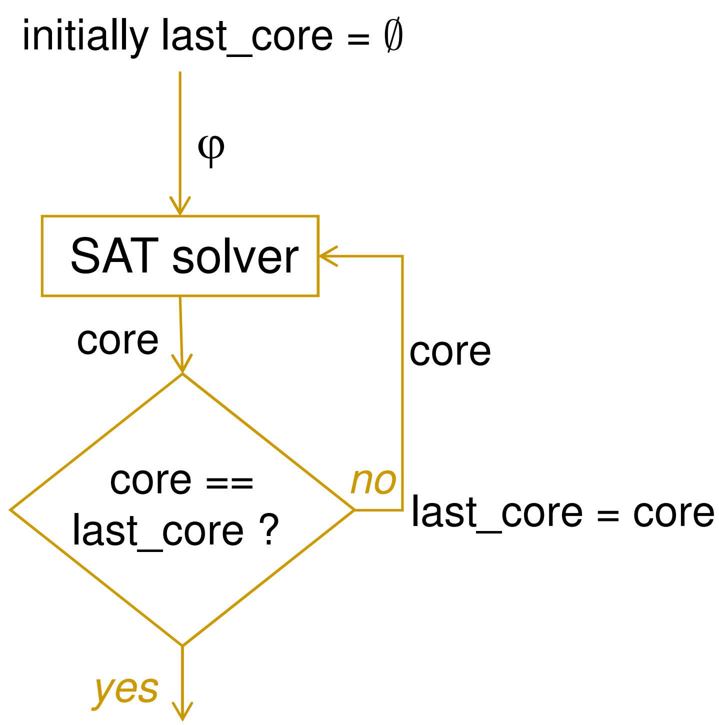
Minimizing the core

- Core compression
  - Smaller core [ZM03, …]
  - Minimal core [DHN06, …]
  - Min-core-biased search [NRS'13]

- Proof compression:
  - Exponential-time transformations [GKS'06]
  - Linear time transformations
    - "Recycle pivots" [BFHSS'08], …

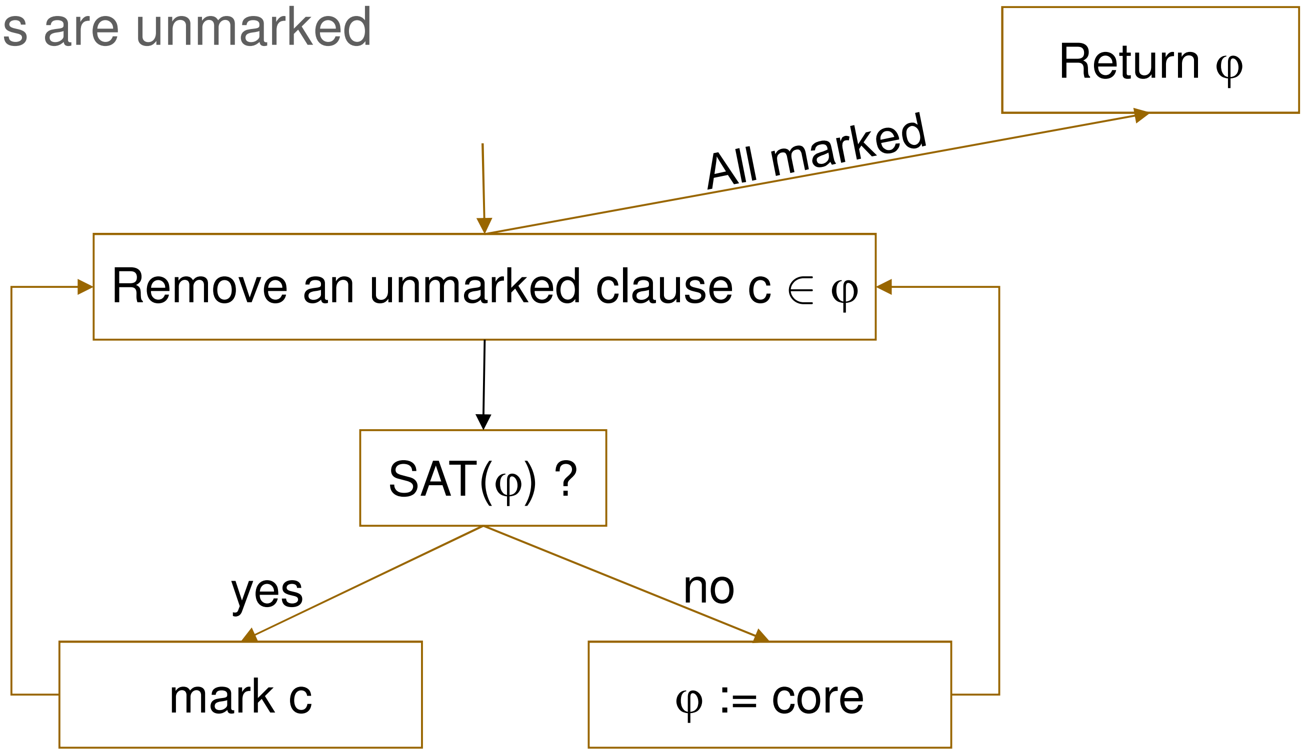
Core compression (smaller core)

- A basic approach: run until reaching a fixpoint [chaff]



Core compression (<u>minimal</u> core)

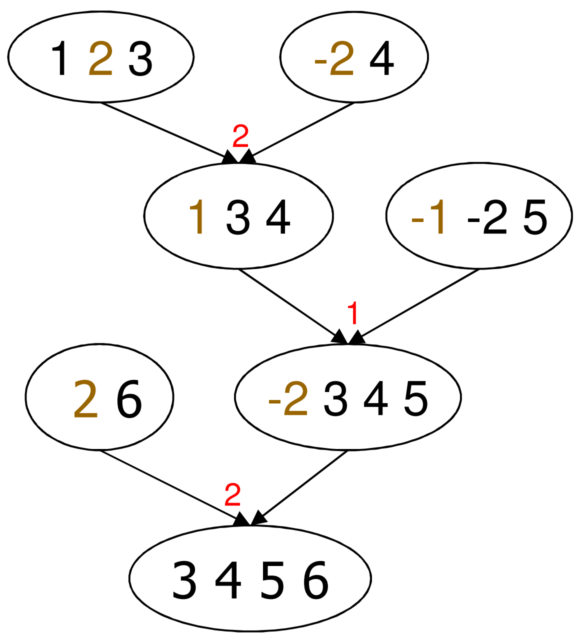
Initially φ's clauses are unmarked



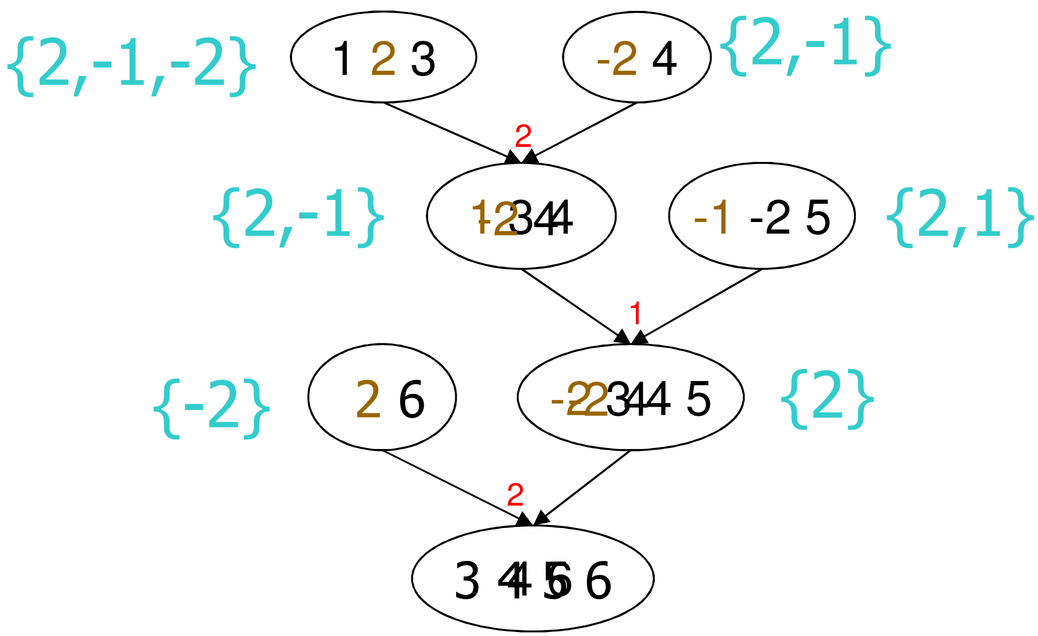
Proof-compression
linear-time transformation / "Recycle-pivots"

- Based on the following fact:
  - Every resolution proof can be made 'regular'
  - … which means that each pivot appears not more than once on every path.

Proof-compression
linear-time transformation / "Recycle-pivots"



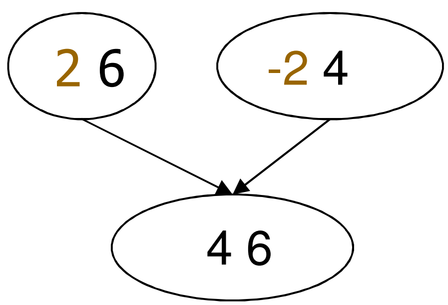
Proof-compression
linear-time transformation / “Recycle-pivots”



Reconstruct proof

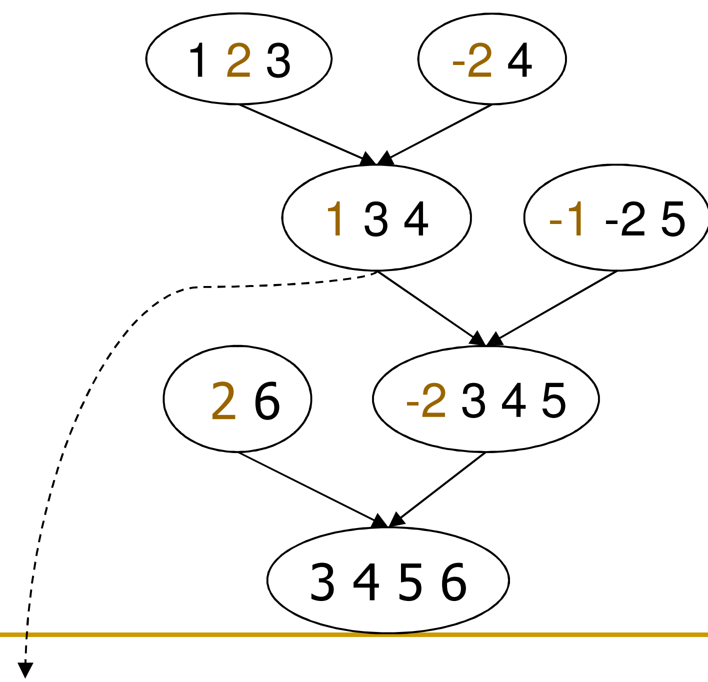
Collect “removable literals”

Proof-compression
linear-time transformation / "Recycle-pivots"



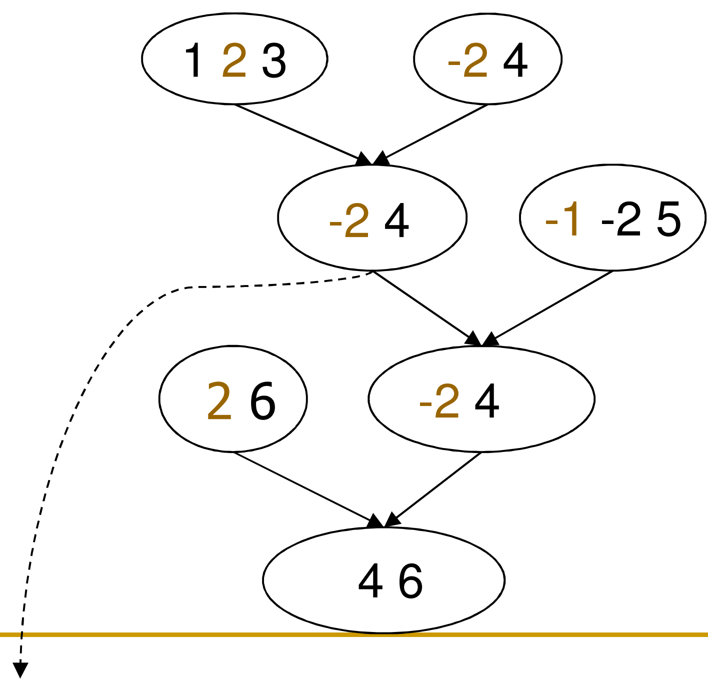
Proof-compression
linear-time transformation / "Recycle-pivots"

- Resolution graphs are DAGs
  - So, a node is on more than one path to the empty clause

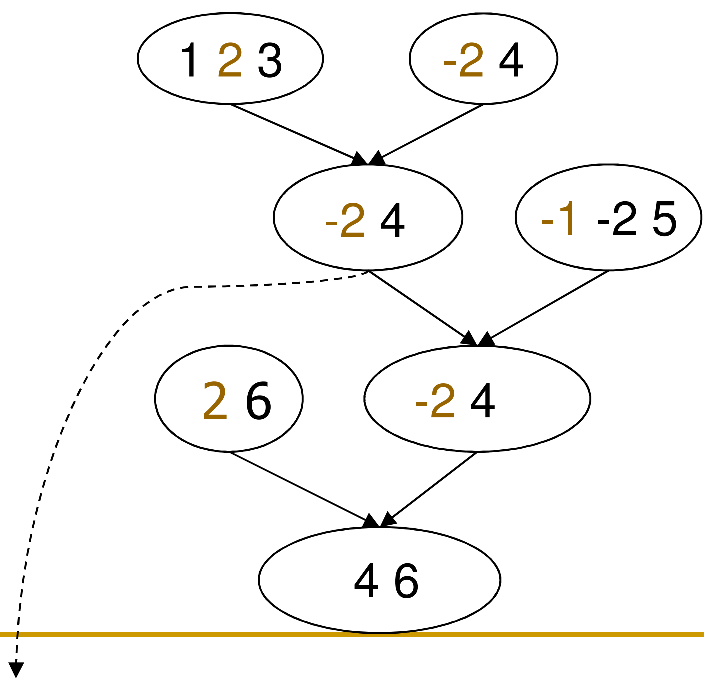


Proof-compression
linear-time transformation / "Recycle-pivots"

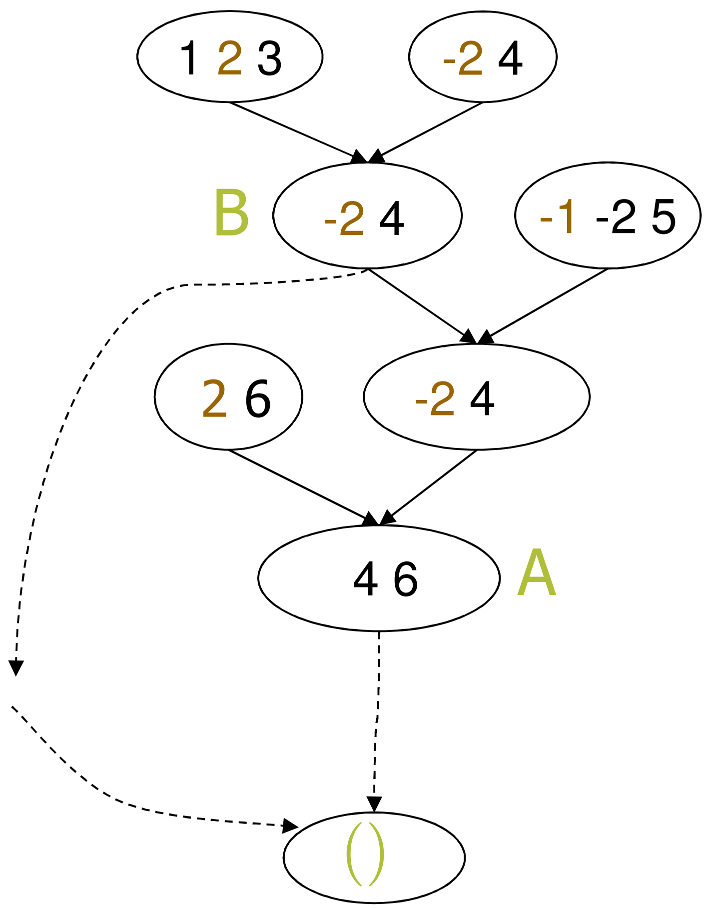
- Resolution graphs are DAGs
  - So, a node is on more than one path to the empty clause



Proof-compression
linear-time transformation / “Recycle-pivots”



Proof-compression
linear-time transformation / “Recycle-pivots”



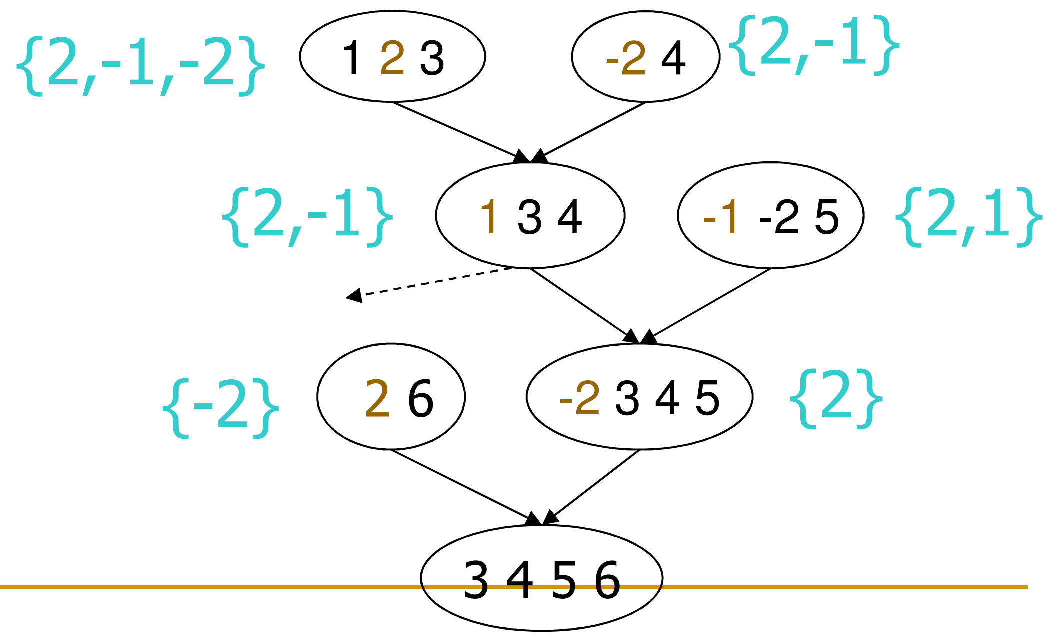
Does A dominate B ?

Dominance relation can be found in O(|E| log |V|)

Problem: need to be updated each time.

Proof-compression
linear-time transformation / "Recycle-pivots"

- Possible solution:
  - Stop propagating information across nodes with more than one child.



Proof-compression
linear-time transformations /recent advances

- Recycle pivots with intersection
  - P. Fontaine, S. Merz and B. W.Paleo. *Compression of Propositional Resolution Proofs via Partial Regularization*. In CADE'11.
- Local transformation Framework
  - R. Bruttomesso, S.F. Rollini, N. Sharygina, and A. Tsitovich. *Flexible Interpolation with Local Proof Transformations*. In ICCAD'10.
  - S.F. Rollini, R. Bruttomesso and N. Sharygina. *An Efficient and Flexible Approach to Resolution Proof Reduction*. In HVC'10.
- Lower units
  - P. Fontaine, S. Merz and B. W.Paleo. *Compression of Propositional Resolution Proofs via Partial Regularization*. In CADE'11.
- Structural hashing
  - S. Cotton. *Two Techniques for Minimizing Resolution Proofs*. In SAT'10.

All implemented in PeRIPLO by S. Rollini. Together they reduce the proof size by ~40%.

Summary

- SAT and CSP are not only about finding models
  - They can provide proofs

- Proofs are important for
  - validation
  - extracting cores
  - various formal-verification techniques

- Minimizing proofs/cores is a subject for intense research.

Questions ?