Satisfiability Modulo Theories and Network Verification

Nikolaj Bjørner
Microsoft Research

Formal Methods and Networks Summer School
Ithaca, June 10-14 2013

Lectures

Wednesday 2:00pm-2:45pm:

An Introduction to SMT with Z3

Thursday 11:00am-11:45am

Algorithmic underpinnings of SAT/SMT

Friday 9:00am-9:45am

Theories, Solvers and **Applications**

Plan

1. Progress in automated reasoning

   SAT, Automated Theorem Proving, SMT

1. An abstract account for SMT search (DPLL+T)

2. Integrating Theories

Takeaway: Theorem Proving is cool and beautiful

Symbolic Engines: SAT, FTP and SMT

SAT: Propositional Satisfiability.

$(Tie \vee Shirt) \wedge (\neg Tie \vee \neg Shirt) \wedge (\neg Tie \vee Shirt)$

FTP: First-order Theorem Proving.

$\forall X,Y,Z\ [X*(Y*Z) = (X*Y)*Z]$

$\forall X\ [X*inv(X) = e]\ \forall X\ [X*e = e]$

SMT: Satisfiability Modulo background Theories

$b + 2 = c \wedge A[3] \neq A[c-b+1]$

SAT - Milestones

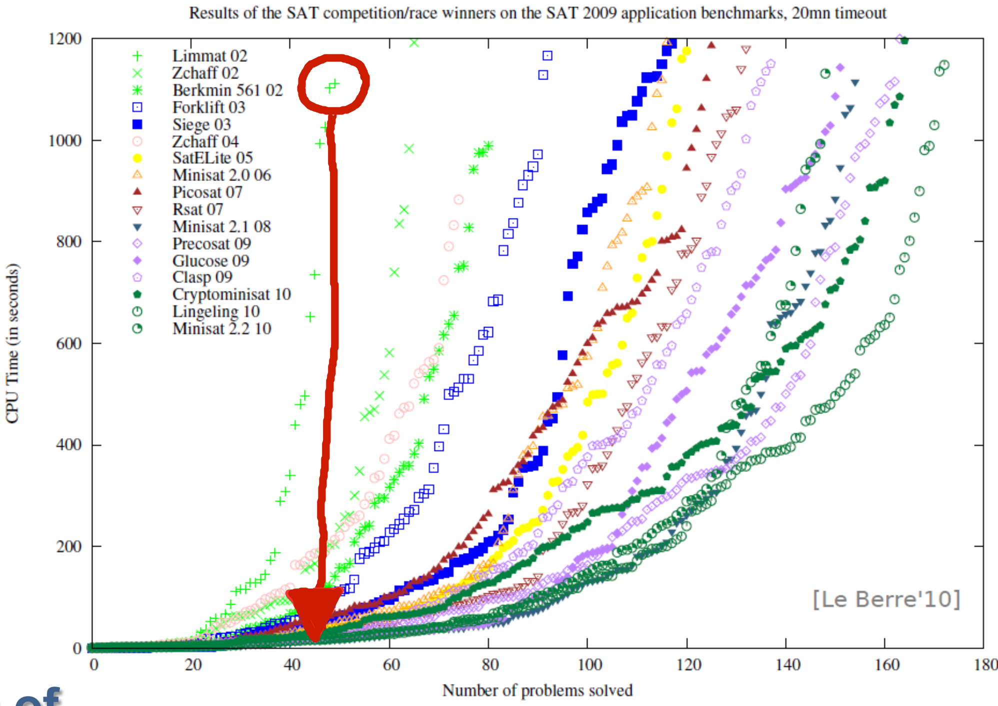
year	Milestone
1960	Davis-Putnam procedure
1962	Davis-Logeman-Loveland
1984	Binary Decision Diagrams
1992	DIMACS SAT challenge
1994	SATO: clause indexing
1997	GRASP: conflict clause learning
1998	Search Restarts
2001	zChaff: 2-watch literal, VSIDS
2005	Preprocessing techniques
2007	Phase caching
2008	Cache optimized indexing
2009	In-processing, clause management
2010	Blocked clause elimination

Problems impossible 10 years ago are trivial today

Concept

2002

2010



Millions of variables from HW designs

Courtesy Daniel le Berre

FTP - Milestones

Year	Milestone	Who	Year	Milestone	Who
1930	Hebrand's theorem	Herbrand	1970	Completion and saturation procedures	many people and provers
1934	Sequent calculi	Gentzen	1970	Knuth-Bendix ordering	Knuth; Bendix
1934	Inverse method	Gentzen	1971	Selection function	Kowalski; Kuehner
1955	Semantic tableaux	Beth	1972	Built-in equational theories	Plotkin
1960	Herbrand-based theorem proving	Wang Hao	1972	Prolog	Colmerauer
1960	Ordered resolution	Davis; Putnam	1974	Saturation algorithms	Overbeek
1962	DLL	Davis; Logemann; Loveland	1975	Completeness of paramodulation	Brand
1963	First-order inverse method	Maslov	1975	AC-unification	Stickel
1965	Unification	J. Robinson	1976	Resolution as a decision procedure	Joyner
1965	First-order resolution	J. Robinson	1979	Basic paramodulation	Degtyarev
1965	Subsumption	J. Robinson	1980	Lexicographic path orderings	Kamin; Levy
1967	Orderings	Slagle	1985	Theory resolution	Stickel
1967	Demodulation or rewriting	Wos; G. Robinson; Carson; Shalla	1986	Definitional clause form transformation	Plaisted; Greenbaum
1968	Model elimination	Loveland	1988	Superposition	Zhang
1969	Paramodulation	G. Robinson; Wos	1988	Model construction	Zhang
			1989	Term indexing	Stickel; Overbeek
			1990	General theory of redundancy	Bachmair; Ganzinger
			1992	Basic superposition	Nieuwenhuis; Rubio
			1993	First instance-based methods	Billon; Plaisted
			1993	Discount saturation algorithm	Avenhaus; Denzinger
			1998	Finite model finding using SAT	McCune
			2000	First-order DPLL	Baumgartner
			2003	iProver method	Ganzinger; Korovin
			2008	Sine selection	Hoder

Some success stories:

- Open Problems (of 25 years): XCB: $X \equiv ((X \equiv Y) \equiv (Z \equiv Y)) \equiv Z)$ is a single axiom for equivalence
- Knowledge Ontologies GBs of formulas

Courtesy Andrei Voronkov, U of Manchester

SMT - Milestones

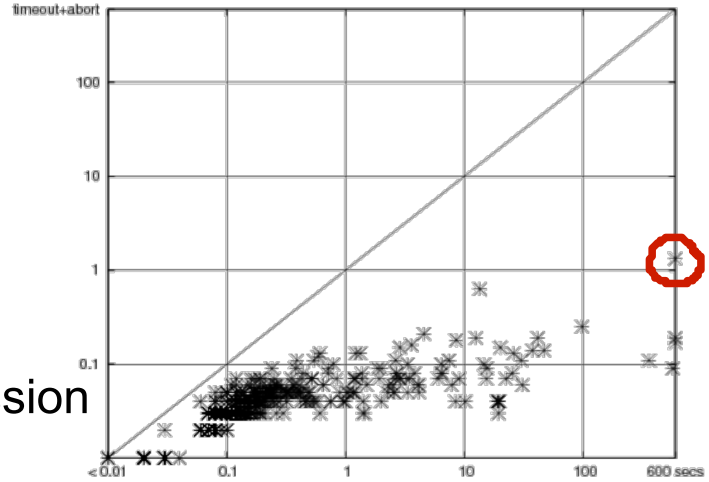
year	Milestone
1977	Efficient Equality Reasoning
1979	Theory Combination Foundations
1979	Arithmetic + Functions
1982	Combining Canonizing Solvers
1992-8	Systems: PVS, Simplify, STeP, SVC
2002	Theory Clause Learning
2005	SMT competition
2006	Efficient SAT + Simplex
2007	Efficient Equality Matching
2009	Combinatory Array Logic, …

Includes progress from SAT:

SAT + Theory Solvers = SMT

15KLOC + 285KLOC = Z3

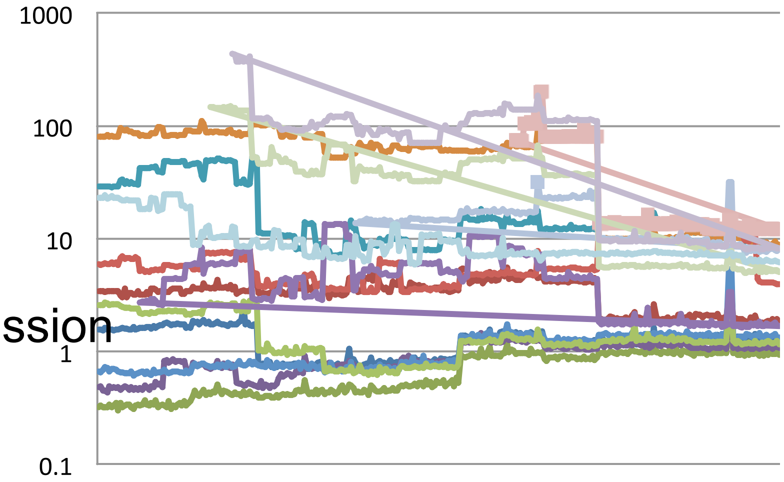
Z3
(of '07)
Time
On
Boogie
Regression



1sec

Simplify (of '01) time

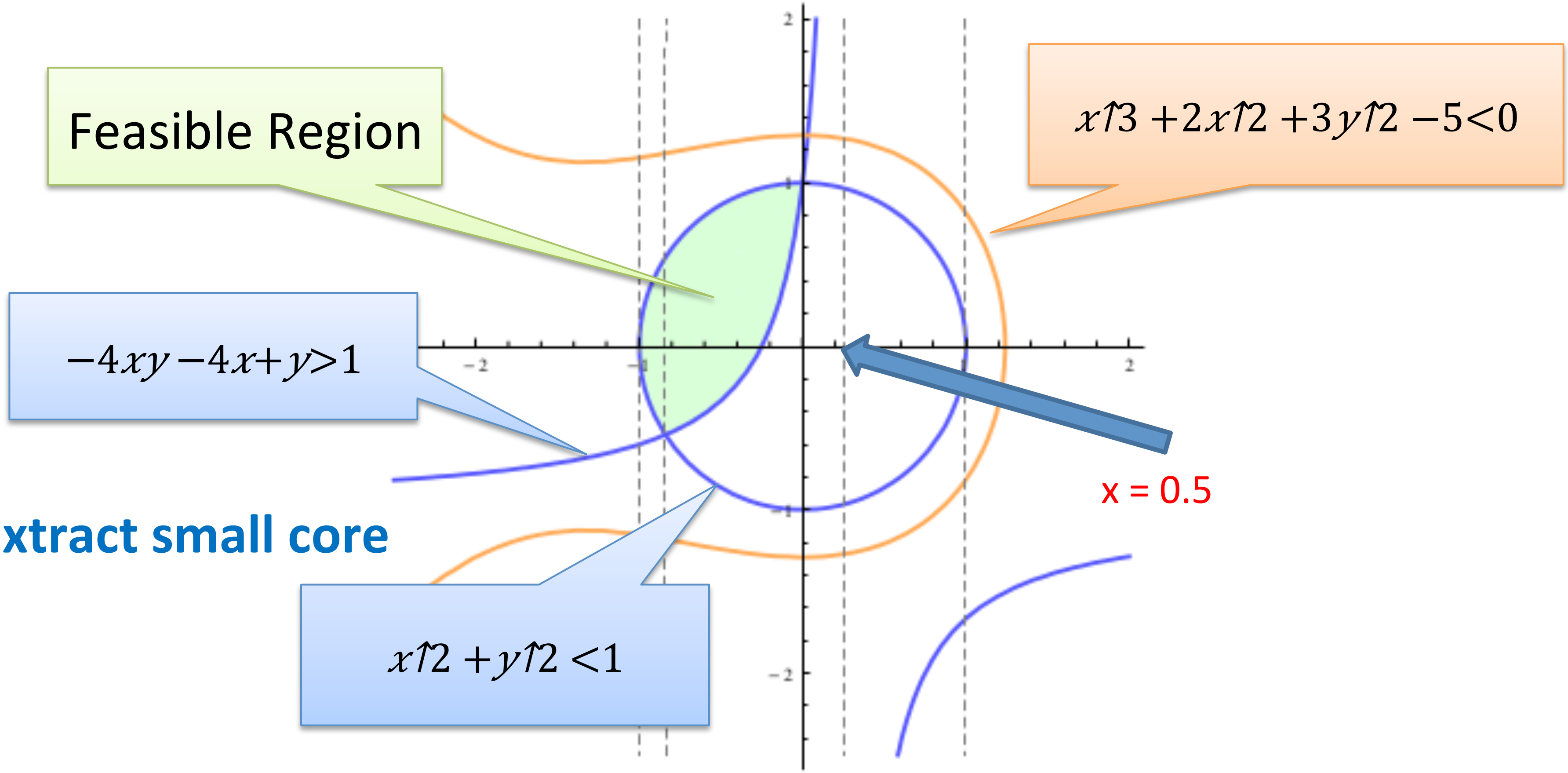
Z3
Time
On
VCC
Regression



Nov 08 March 09

Z3 News: Solving ∃R Efficiently

A key idea: Use partial solution to guide the search



Feasible Region

$x\uparrow 3 + 2x\uparrow 2 + 3y\uparrow 2 - 5 < 0$

$-4xy - 4x + y > 1$

$x\uparrow 2 + y\uparrow 2 < 1$

x = 0.5

Extract small core

Dejan Jojanovich & Leonardo de Moura, IJCAR 2012

News: Horn Clause Satisfiability

mc(x) = x-10 if x > 100

mc(x) = mc(mc(x+11)) if x ≤ 100

assert (x ≤ 101 ⇒ mc(x) = 91)

$$\forall X.\ X > 100 \rightarrow \mathrm{mc}(X, X-10)$$

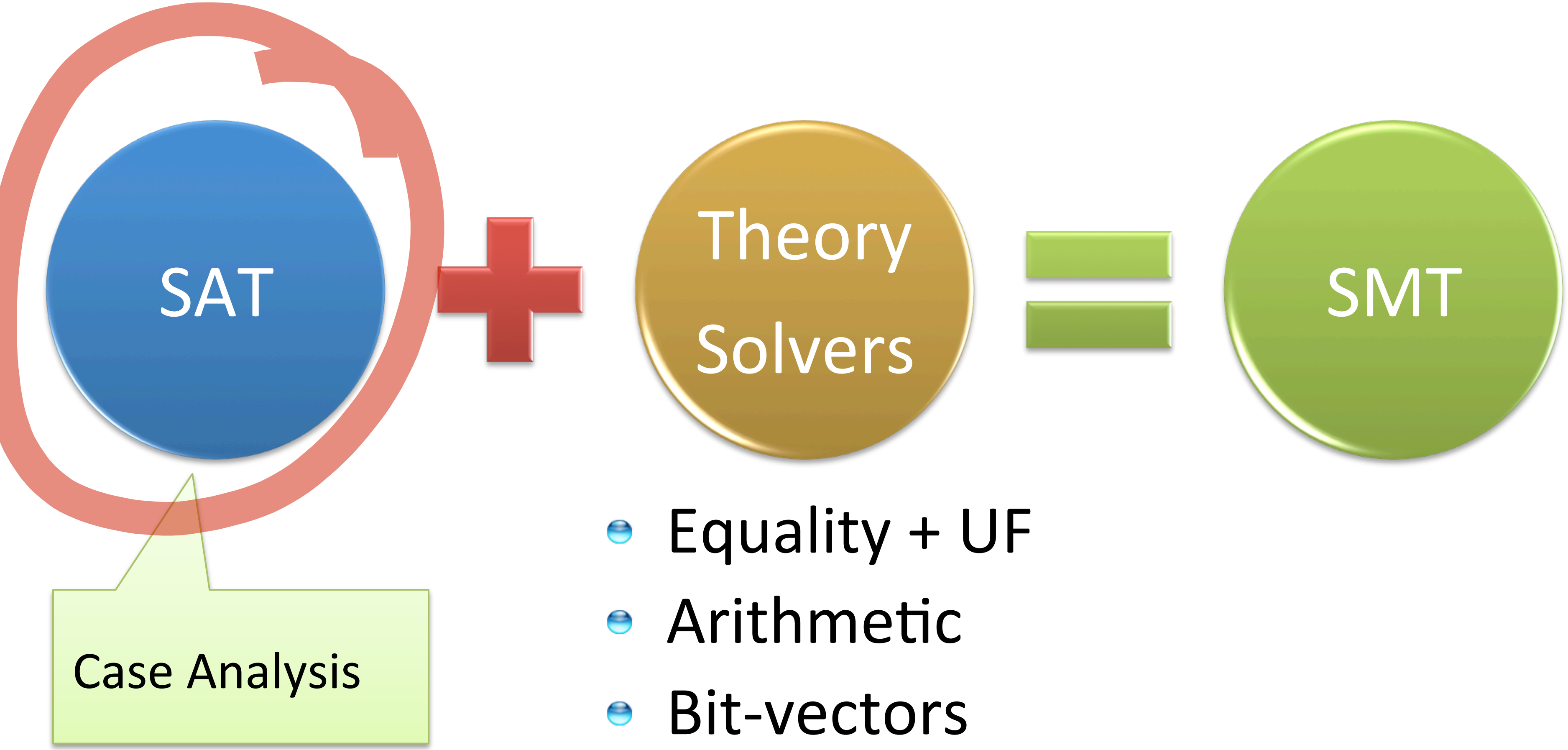
$$\forall X,Y,R.\ X \leq 100 \wedge \mathrm{mc}(X+11, Y) \wedge \mathrm{mc}(Y, R) \rightarrow \mathrm{mc}(X, R)$$

$$\forall X,R.\ \mathrm{mc}(X, R) \wedge X \leq 101 \rightarrow R = 91$$

Krystof Hoder & Nikolaj Bjorner, SAT 2012
Bjorner, McMillan, Rybalchenko, SMT 2012

SMT SOLVING

SMT : Basic Architecture



Case Analysis

- Equality + UF
- Arithmetic
- Bit-vectors
- …

SAT + Theory solvers

Basic Idea

$x \geq 0,\ y = x + 1,\ (y > 2 \lor y < 1)$

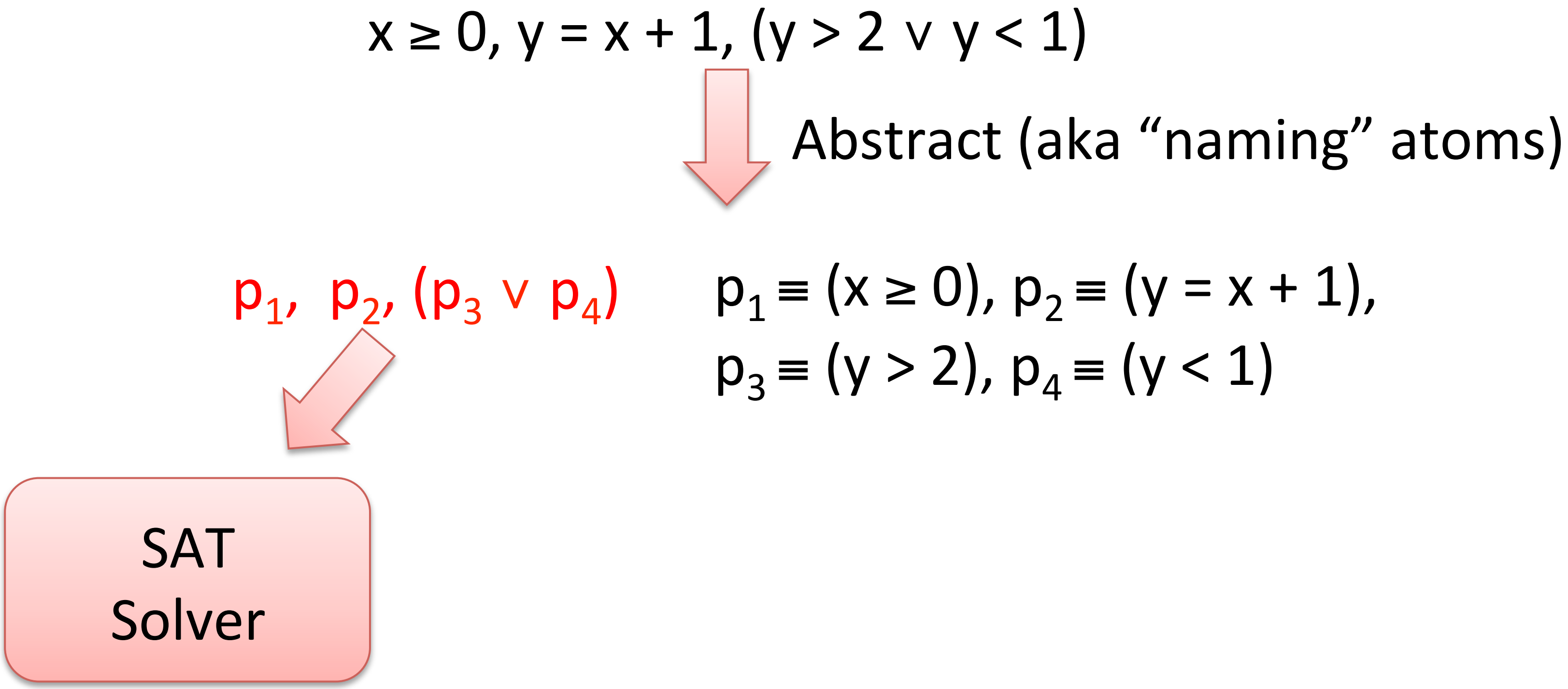
Abstract (aka “naming” atoms)

$p_1,\ p_2,\ (p_3 \lor p_4)$

$p_1 \equiv (x \geq 0),\ p_2 \equiv (y = x + 1),$
$p_3 \equiv (y > 2),\ p_4 \equiv (y < 1)$

SAT + Theory solvers

Basic Idea



$x \geq 0, y = x + 1, (y > 2 \vee y < 1)$

Abstract (aka “naming” atoms)

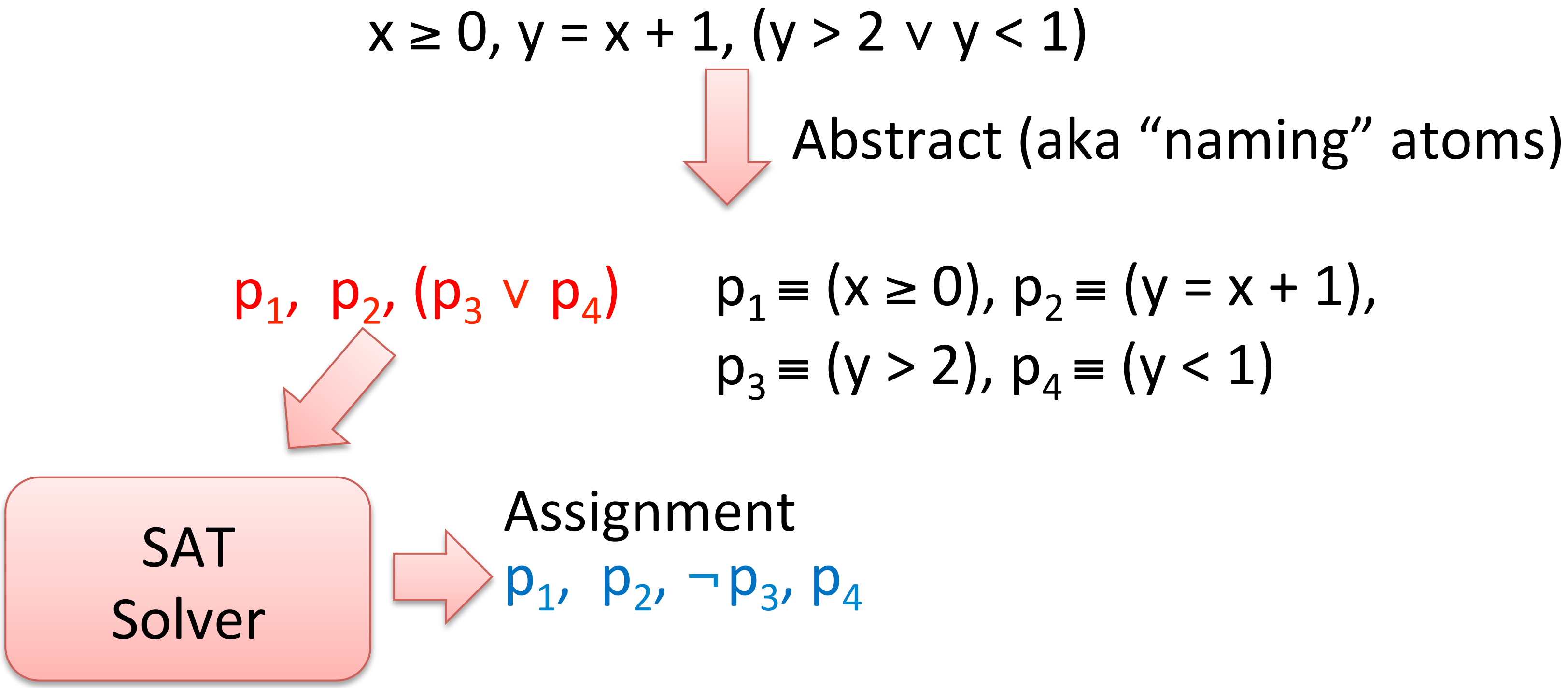
$p_1, p_2, (p_3 \vee p_4)$

$p_1 \equiv (x \geq 0), p_2 \equiv (y = x + 1),$
$p_3 \equiv (y > 2), p_4 \equiv (y < 1)$

SAT Solver

SAT + Theory solvers

Basic Idea



$x \geq 0,\ y = x + 1,\ (y > 2 \lor y < 1)$

Abstract (aka “naming” atoms)

$p_1,\ p_2,\ (p_3 \lor p_4)$

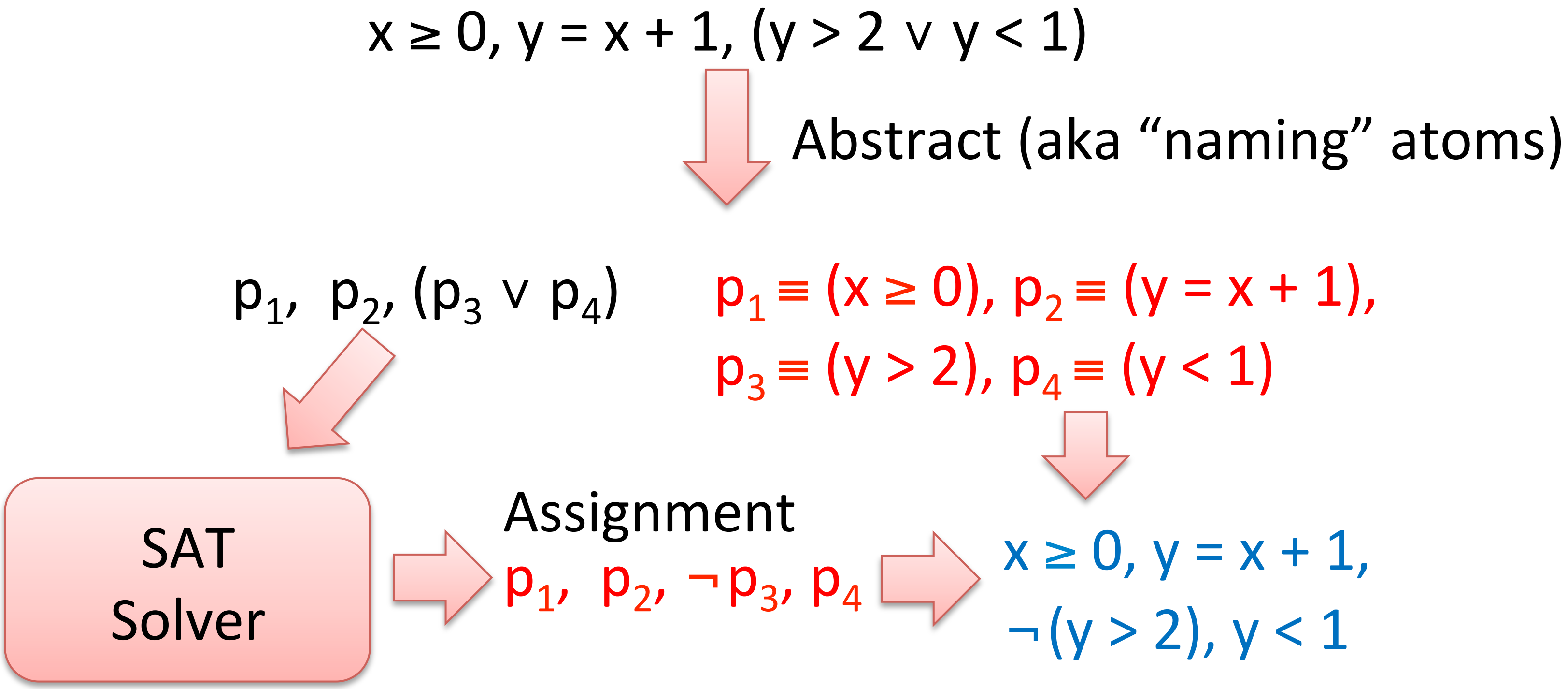
$p_1 \equiv (x \geq 0),\ p_2 \equiv (y = x + 1),$
$p_3 \equiv (y > 2),\ p_4 \equiv (y < 1)$

SAT Solver

Assignment
$p_1,\ p_2,\ \neg p_3,\ p_4$

SAT + Theory solvers

Basic Idea



$x \geq 0, y = x + 1, (y > 2 \vee y < 1)$

Abstract (aka “naming” atoms)

$p_1, \; p_2, (p_3 \vee p_4)$

$p_1 \equiv (x \geq 0), p_2 \equiv (y = x + 1),$
$p_3 \equiv (y > 2), p_4 \equiv (y < 1)$

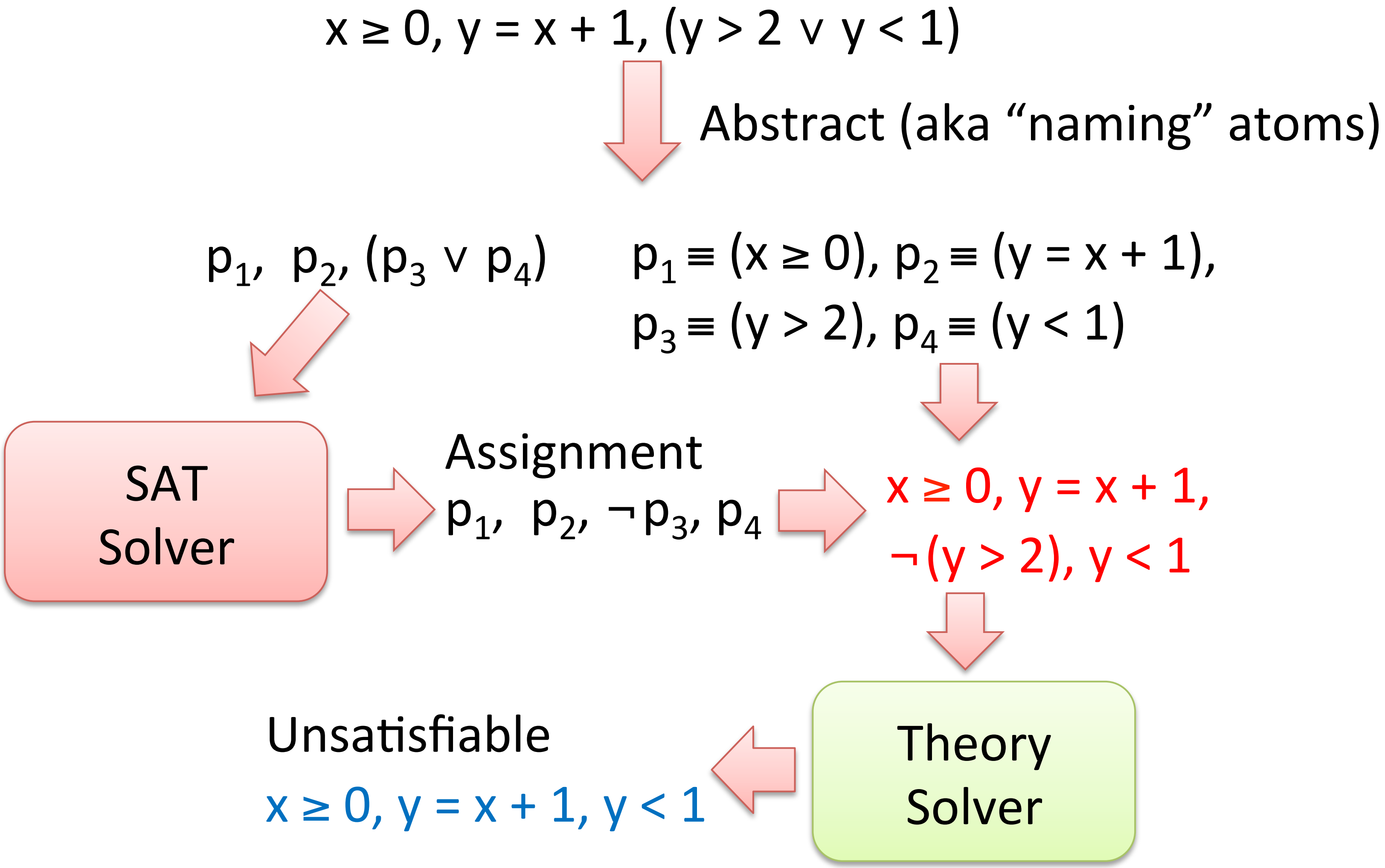
SAT Solver

Assignment
$p_1, \; p_2, \neg p_3, p_4$

$x \geq 0, y = x + 1,$
$\neg (y > 2), y < 1$

SAT + Theory solvers

Basic Idea



$x \geq 0, y = x + 1, (y > 2 \vee y < 1)$

Abstract (aka “naming” atoms)

$p_1, p_2, (p_3 \vee p_4)$

$p_1 \equiv (x \geq 0), p_2 \equiv (y = x + 1),$
$p_3 \equiv (y > 2), p_4 \equiv (y < 1)$

SAT Solver

Assignment
$p_1, p_2, \neg p_3, p_4$

$x \geq 0, y = x + 1,$
$\neg (y > 2), y < 1$

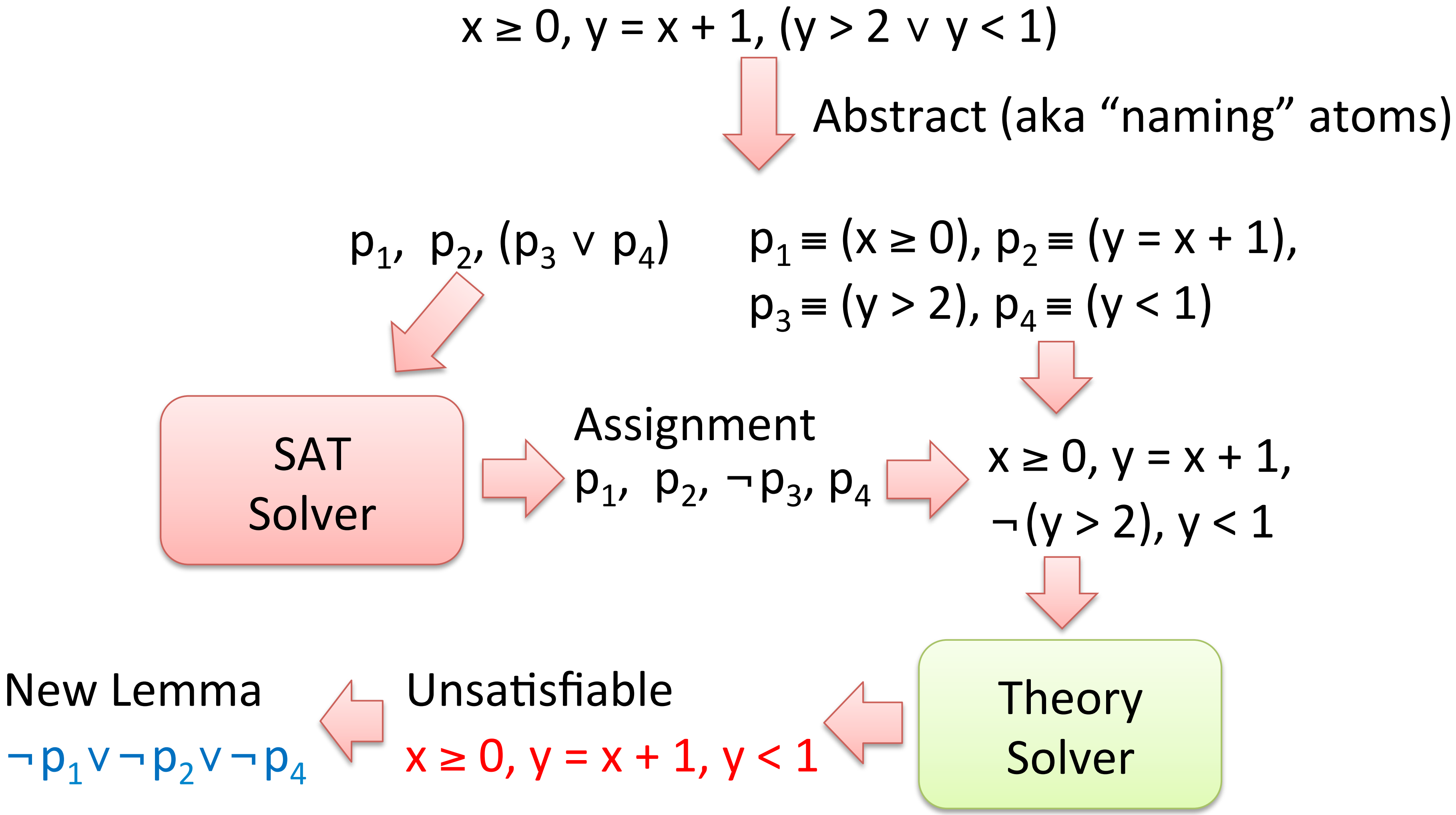
Theory Solver

Unsatisfiable

$x \geq 0, y = x + 1, y < 1$

SAT + Theory solvers

Basic Idea



$x \geq 0, y = x + 1, (y > 2 \vee y < 1)$

Abstract (aka "naming" atoms)

$p_1, p_2, (p_3 \vee p_4)$

$p_1 \equiv (x \geq 0), p_2 \equiv (y = x + 1),$
$p_3 \equiv (y > 2), p_4 \equiv (y < 1)$

SAT Solver

Assignment
$p_1, p_2, \neg p_3, p_4$

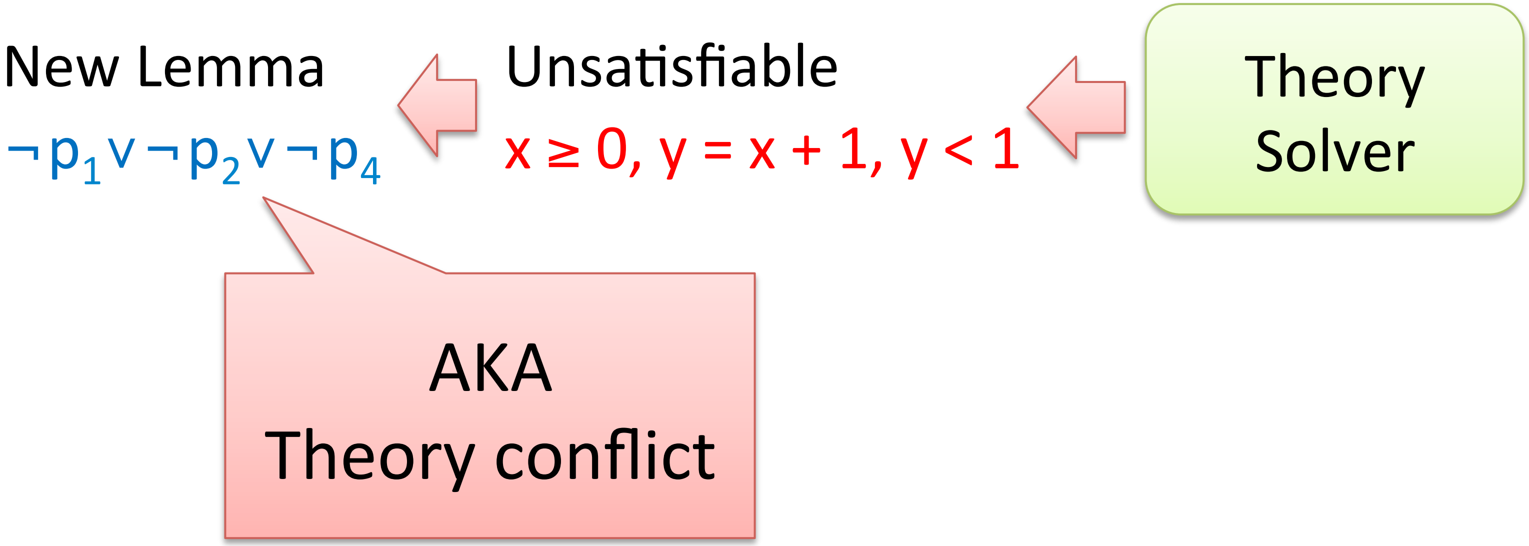
$x \geq 0, y = x + 1,$
$\neg(y > 2), y < 1$

Theory Solver

Unsatisfiable
$x \geq 0, y = x + 1, y < 1$

New Lemma
$\neg p_1 \vee \neg p_2 \vee \neg p_4$

SAT + Theory solvers



New Lemma

$\neg p_1 \vee \neg p_2 \vee \neg p_4$

Unsatisfiable

$x \geq 0, y = x + 1, y < 1$

Theory Solver

AKA

Theory conflict

SAT/SMT SOLVING USING DPLL(T)

[DAVIS PUTNAM LOGEMAN LOVELAND MODULO THEORIES]

Resolution

Formula must be in CNF

Resolution rule: $C \vee p \qquad D \vee \neg p / C \vee D$

Example: $q \vee t \vee p \qquad q \vee r \vee \neg p / q \vee t \vee r$

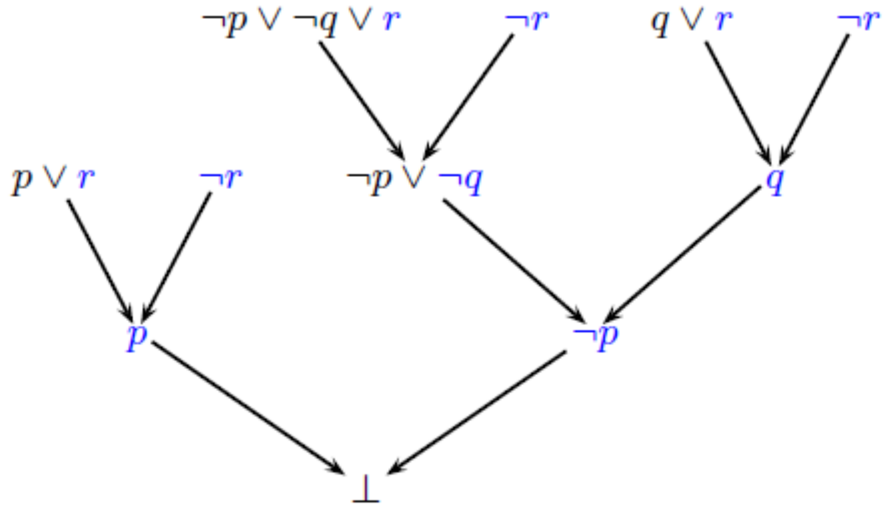
The result of resolution is the resolvent (clause).
Original clauses are kept (not deleted).
Duplicate literals are deleted from the resolvent.

Note: No branching.

Termination: Only finite number of possible derived clauses.

Resolution (example)

A refutation of $\neg p \vee \neg q \vee r$, $p \vee r$, $q \vee r$, $\neg r$:



Ex: Implement a naïve resolution procedure.

Unit & Input Resolution

Unit resolution: $C \vee \ell \quad \neg\ell / C \quad \neg\ell$ ($C \vee \ell$ is subsumed by C)

Input resolution: $C \vee \ell \quad D \vee \neg\ell / C \vee D$ ($C \vee \ell$ member of input *F*).

Exercise:

Set of clauses *F:*

F has an input refutation iff *F* has a unit refutation.

DPLL

DPLL: David Putnam Logeman Loveland = Unit resolution + split rule.

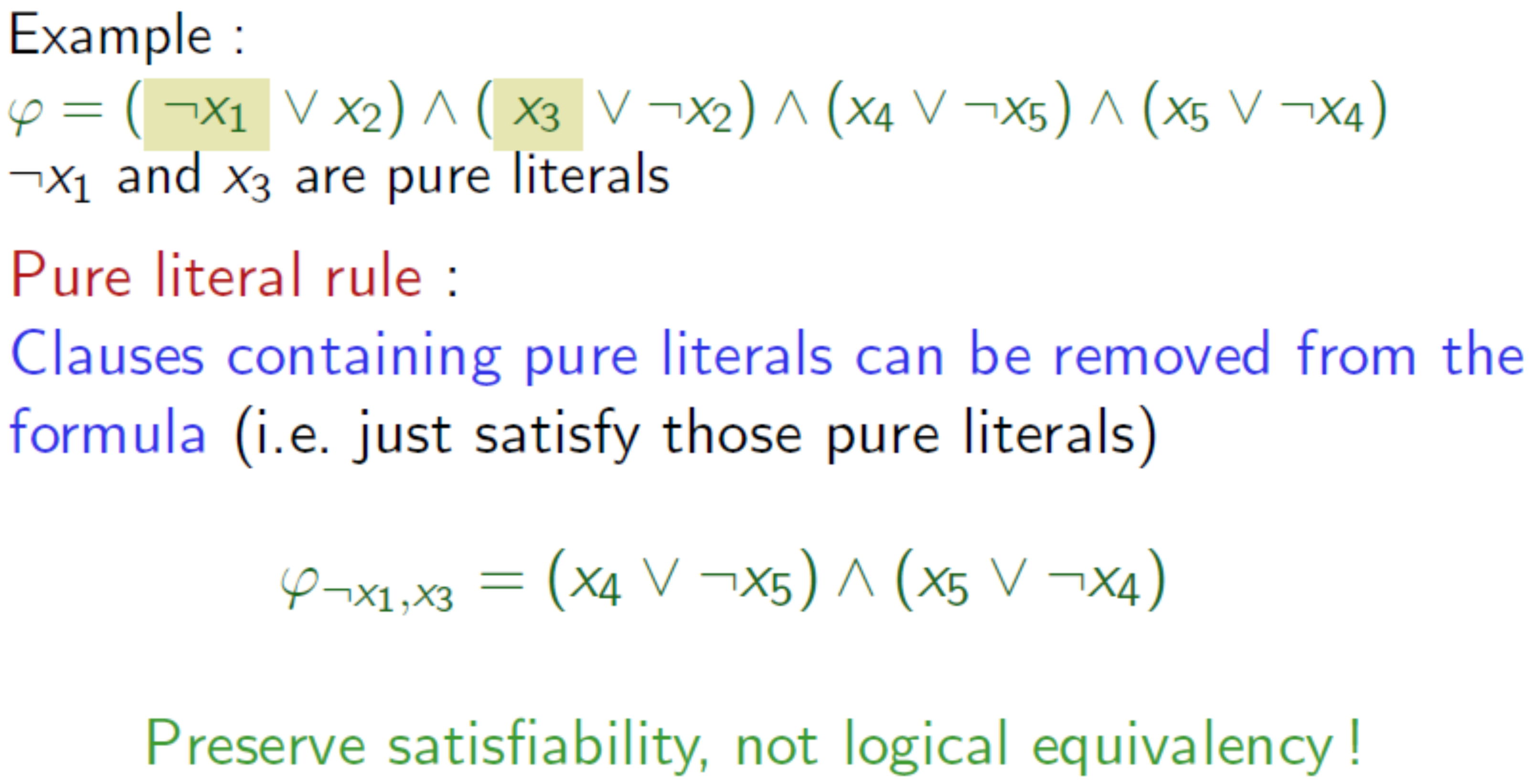
$F / F, p \quad | \quad F, \neg p$ split $\quad$ *p and ¬p are not in F*

$F, \quad C \vee \ell \,, \neg\ell \,/ F, \; C, \; \neg\ell$ unit

Ingredient of most efficient SAT solvers

Pure Literals

A literal is pure if only occurs positively or negatively.



Example :

$\varphi = (\neg x_1 \lor x_2) \land (x_3 \lor \neg x_2) \land (x_4 \lor \neg x_5) \land (x_5 \lor \neg x_4)$

$\neg x_1$ and x_3 are pure literals

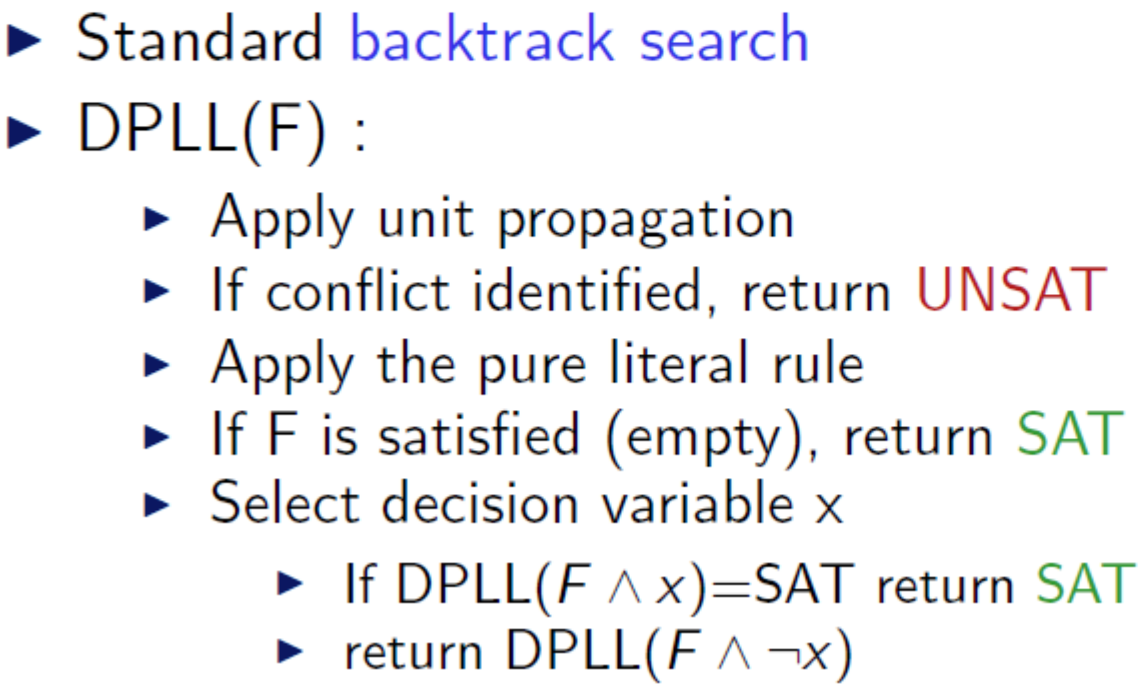
Pure literal rule :

Clauses containing pure literals can be removed from the formula (i.e. just satisfy those pure literals)

$$\varphi_{\neg x_1, x_3} = (x_4 \lor \neg x_5) \land (x_5 \lor \neg x_4)$$

Preserve satisfiability, not logical equivalency !

DPLL (as a procedure)



- Standard backtrack search
- DPLL(F) :
  - Apply unit propagation
  - If conflict identified, return UNSAT
  - Apply the pure literal rule
  - If F is satisfied (empty), return SAT
  - Select decision variable x
    - If DPLL($F \wedge x$)=SAT return SAT
    - return DPLL($F \wedge \neg x$)

DPLL

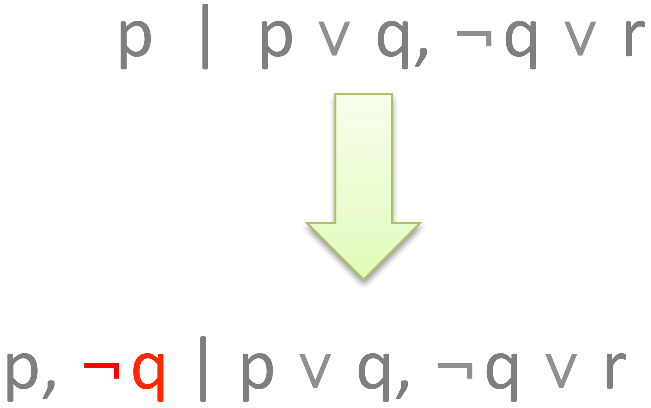
M | F

Partial model

Set of clauses

DPLL

Guessing



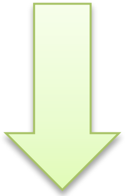
$p \mid p \vee q, \neg q \vee r$

$p, \neg q \mid p \vee q, \neg q \vee r$

DPLL

Deducing

$p \mid p \vee q, \neg p \vee s$



$p, s \mid p \vee q, \neg p \vee s$

DPLL

Backtracking

$p, \neg s, q \mid p \vee q, s \vee q, \neg p \vee \neg q$

↓

$p, s \mid p \vee q, s \vee q, \neg p \vee \neg q$

Modern DPLL

- Non-chronological backtracking (backjumping)
- Lemma learning

and

- Efficient indexing (two-watch literal)
- …

CDCL – Conflict Directed Clause Learning

Lemma learning

$\neg t, p, q, s \mid t \vee \neg p \vee q, \neg q \vee s, \neg p \vee \neg s$

$\neg t, p, q, s \mid t \vee \neg p \vee q, \neg q \vee s, \neg p \vee \neg s \mid \neg p \vee \neg s$

$\neg t, p, q, s \mid t \vee \neg p \vee q, \neg q \vee s, \neg p \vee \neg s \mid \neg p \vee \neg q$

$\neg t, p, q, s \mid t \vee \neg p \vee q, \neg q \vee s, \neg p \vee \neg s \mid \neg p \vee t$

Core Engine in Z3: Modern DPLL/CDCL

Initialize	$\epsilon \mid F$	
Decide	$M \mid F \Longrightarrow M, \ell \mid F$	
Propagate	$M \mid F, C \vee \ell \Longrightarrow M, \ell^{C \vee \ell} \mid F, C \vee \ell$	
Sat	$M \mid F \Longrightarrow M$	
Conflict	$M \mid F, C \Longrightarrow M \mid F, C \mid C$	C is f… der M
	$C \Longrightarrow M \mid F, C \mid C$	
	$\Longrightarrow Unsat$	
	$\vee \ell \Longrightarrow M \ell^{C \vee \ell} \mid F$	$C \subseteq M, \neg \ell \in M'$
	$\mid C' \vee \neg \ell \Longrightarrow M \mid F \mid C' \vee C$	$\ell^{C \vee \ell} \in M$
orget	$M \mid F, C \Longrightarrow M \mid F$	C is a learned clause
Restart	$M \mid F \Longrightarrow \epsilon \mid F$	

roof

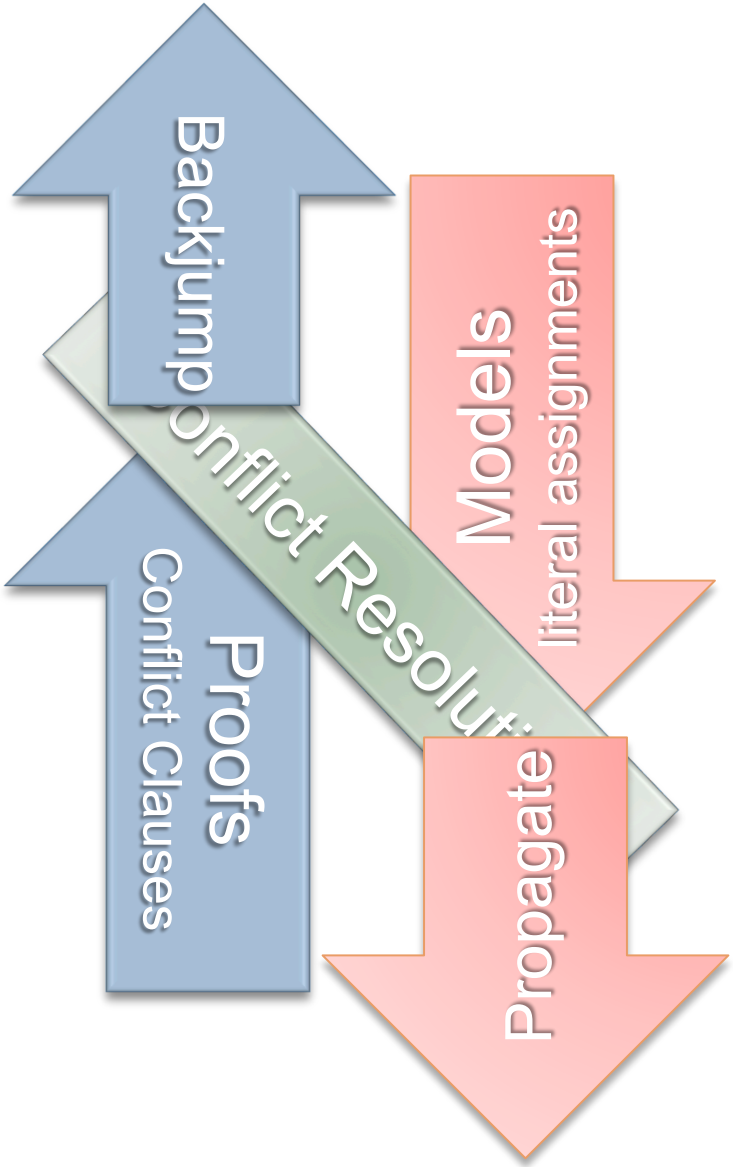
Conflict Resolution

"It took me a year to understand the Mini-SAT FUIP code"
Mate Soos to
Niklas Sörenson
over ice-cream in Trento

We will **now** motivate the CDCL algorithm as a cooperative procedure between model and proof search

[Nieuwenhuis, Oliveras, Tinelli J.ACM 06] customized

Mile High: Modern SAT/SMT search



The Farkas Lemma Dichotomy

1. There is an x such that: $Ax = b \wedge x \geq 0$

2. There is a y such that: $yA \geq 0 \wedge yb < 0$

For every matrix A, vector b it is the case that either (1) or (2) holds (and not both).

A Dichotomy of Models and Proofs

1. There is a model *M* such that $M \models F$

2. There is a proof Π such that $F \vdash \downarrow \Pi \ \emptyset$

For every formula *F* (set of clauses) it is the case that either (1) or (2) holds (and not both).

A Dichotomy of Models and Proofs

1. There is $M' \supseteq M$ such that $M' \vDash F$

2. There is $M' \subseteq M$ and proof Π such that $F \vdash \downarrow \Pi \; M'$

For every formula *F* (set of clauses) and partial model M it is the case that either (1) or (2) holds (and not both).

A Dichotomy of Models and Proofs

1. There is $M' \supseteq M$ such that $M' \vDash F$

2. There is $M' \subseteq M$ and proof Π such that $F \vdash \downarrow \Pi \; M'$

Given M can it be extended to M' to satisfy (1)?
If not, find subset M' to establish (2).
(that is inconsistent with *F*)

A Dichotomy of Models and Proofs

Corollary:

If $F \vdash \swarrow \Pi\ C$ then it is not possible to extend C to satisfy F

Corollary:

If $M \vDash \neg F$ then

- $C, \ell \subseteq M$ for some $F \vdash C \vee \ell$ (or F contains ∅)
- for every D, where
    - $D, C \subseteq M{\upharpoonleft}' \subseteq M$,
    - $M{\upharpoonleft}' \vdash (D \vee \neg \ell)$

  it is not possible to extend $M{\upharpoonleft}'$ to satisfy F

CDCL Search – Data structures

$M \mid F$

Partial Model:
Sequence of literals
Decision lits:
case splits
Propagation lits:
only one case
makes sense.

Formula:
set of clauses

Proof: Implicit
Consequences added to *F*

Invariant:

For state $M \mid F \mid C$: $C \subseteq M \quad F \vdash C$

Invariant:

For states $M \mid F$ and $M \mid F \mid D$ where $M = M_1\, \ell^{C \vee \ell}\, M_2$: $C \subseteq M_1 \quad F \vdash C \vee \ell$

CDCL steps

Initialize $\epsilon | F$ $F \text{ is a set of clauses}$

No model candidate has been fixed

CDCL steps

Decide $M \mid F \Longrightarrow M, \ell \ \mid F$ ℓ *is unassigned*

Case split on ℓ
If Mcan be extended to satisfy F,
then the extension contains M, p or $M, \neg p$

CDCL steps

Propagate $M \mid F, C \vee \ell \Longrightarrow M, \ell \uparrow C \vee \ell \quad \mid F, C \vee \ell$ $C\ is\ false\ under\ M$

ℓ must be true if M has any chance of being a model for $F, C \vee \ell$

CDCL steps

Sat	$M \mid F \Longrightarrow M$	$F\ true\ under\ M$
Unsat	$M \mid F \mid \emptyset \Longrightarrow Unsat$	

CDCL steps

Conflict $M \mid F, C \Longrightarrow M \mid F, C \mid C$ $C\ is\ false\ under\ M$

C is a **sufficient** explanation why M is not a model of F

CDCL steps

Resolve $M \mid F \mid C\vee\neg\ell \Longrightarrow M \mid F \mid C\vee D$ $\ell \uparrow D\vee\ell \in M$

Recall

Corollary:
If $M \vDash \neg F$ then
- $C,\ell \subseteq M$ for some $F \vdash C\vee\ell$
  (or F contains ∅)
- for every D, where
  - $D, C \subseteq M\upharpoonright \subseteq M$,
  - $M\upharpoonright \vdash (D\vee\neg\ell)$

  it is not possible to extend $M\upharpoonright$ to satisfy F

$C\vee D$ is a sufficient and **earlier** explanation why M is not a model of F

CDCL steps

Backjump $MM' \mid F \mid C \vee \ell \Longrightarrow M \ell^{C \vee \ell} \mid F$ $C \subseteq M, \neg\ell \in M'$

- $C \vee \ell$ is a sufficient explanation why M is not a model of F
- Prefixes of MM' that contain $\neg\ell$ cannot become a model of F

FUIP *First Unique Implication Point* strategy when # of decision literals in M is minimal.

Why is **FUIP** better?

- Minimizes # of backtracking points before learned fact $\ell^{C \vee \ell}$
- What if $\ell^{C \vee \ell}$ implies negation of removed backtracking point?
  - We would *forget* the learned fact $\ell^{C \vee \ell}$ during backjumping.
  - … only to then re-learn it.

CDCL steps

Learn $M \mid F \mid C \Longrightarrow M \mid F, C \mid C$

Re-use proof step for later: build DAG proof instead of TREE proof

CDCL steps

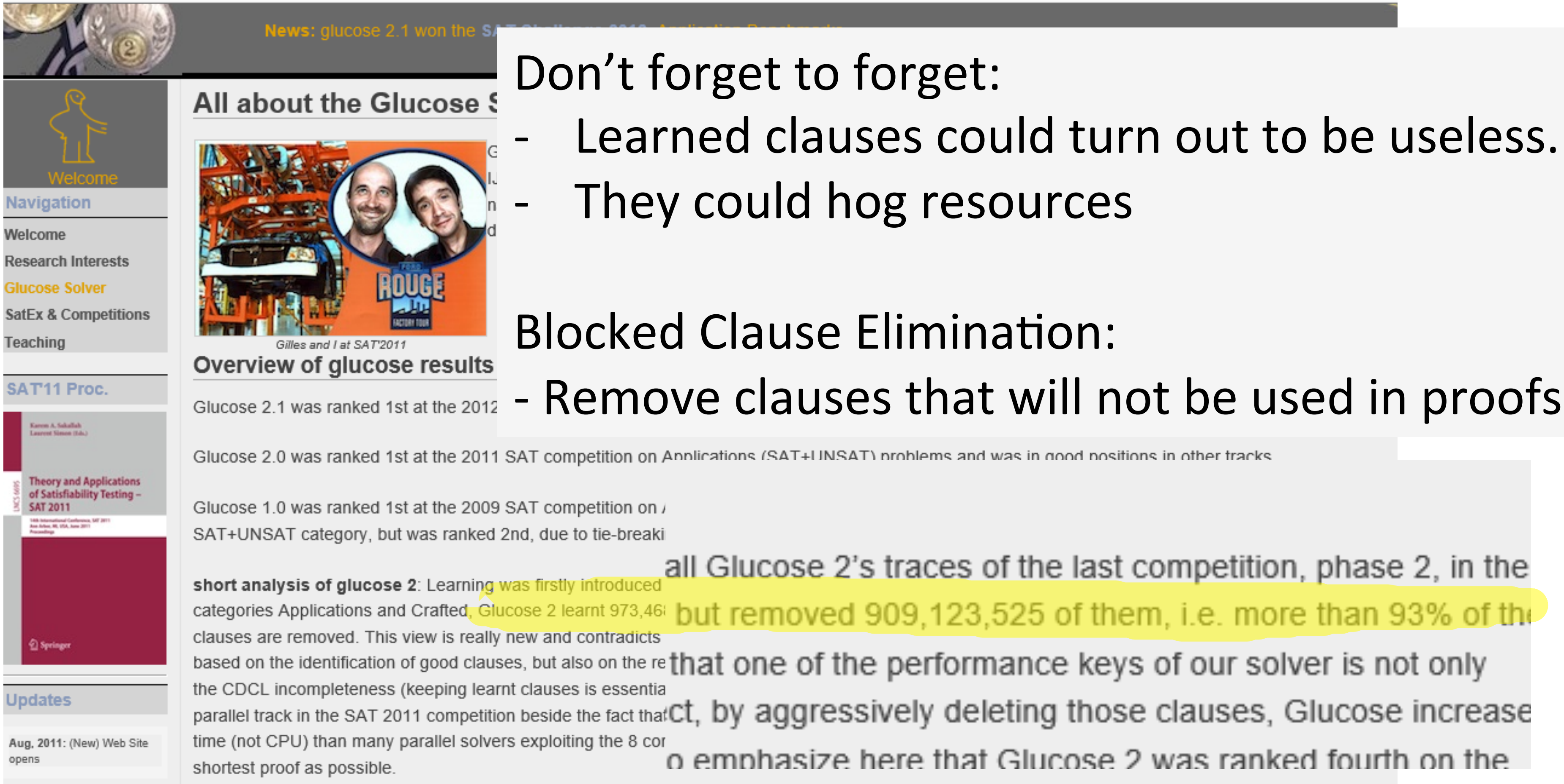
Forget $M \mid F, C \Longrightarrow M \mid F$ C is a learned clause

Don’t forget to forget:

- Learned clauses could turn out to be useless.
- They could hog resources

Blocked Clause Elimination:

- Remove clauses that will not be used in proofs



News: glucose 2.1 won the SA

Welcome

Navigation

Welcome

Research Interests

Glucose Solver

SatEx & Competitions

Teaching

SAT'11 Proc.

Updates

Aug, 2011: (New) Web Site opens

All about the Glucose S

G

IJ

n

d

Gilles and I at SAT'2011

Overview of glucose results

Glucose 2.1 was ranked 1st at the 2012

Glucose 2.0 was ranked 1st at the 2011 SAT competition on Applications (SAT+UNSAT) problems and was in good positions in other tracks

Glucose 1.0 was ranked 1st at the 2009 SAT competition on A
SAT+UNSAT category, but was ranked 2nd, due to tie-breaki

short analysis of glucose 2: Learning was firstly introduced
categories Applications and Crafted, Glucose 2 learnt 973,46
clauses are removed. This view is really new and contradicts
based on the identification of good clauses, but also on the re
the CDCL incompleteness (keeping learnt clauses is essentia
parallel track in the SAT 2011 competition beside the fact that
time (not CPU) than many parallel solvers exploiting the 8 cor
shortest proof as possible.

all Glucose 2's traces of the last competition, phase 2, in the
but removed 909,123,525 of them, i.e. more than 93% of the
that one of the performance keys of our solver is not only
ct, by aggressively deleting those clauses, Glucose increase
o emphasize here that Glucose 2 was ranked fourth on the

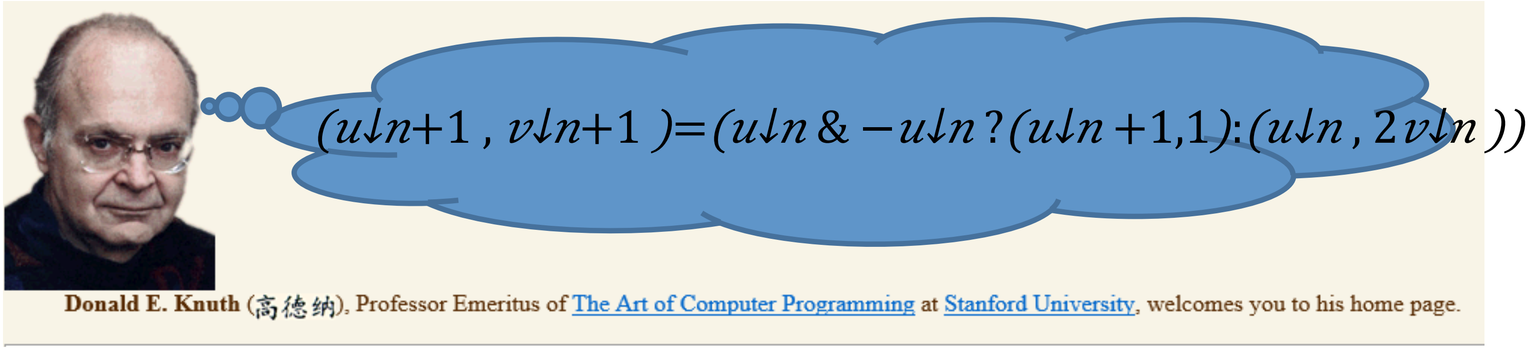
CDCL steps

Restart $M \mid F \Longrightarrow \epsilon \mid F$

Avoid getting trapped in one part of search space

$S_1, S_2, \ldots = 1,1,2,1,1,2,4,1,1,2,1,1,2,4,8,1,1,2,1,1,2,4,1,1,2,4,8,1,\ldots$

[Reluctant doubling sequence: Luby, Sinclair, Zuckerman, IPL 47]



$(u_{n+1}, v_{n+1}) = (u_n \& -u_n ? (u_n+1, 1) : (u_n, 2v_n))$.

Donald E. Knuth (高德纳), Professor Emeritus of The Art of Computer Programming at Stanford University, welcomes you to his home page.

Generating function [fasc6a draft chapter on SAT]

Modern DPLL - tuning

- Restart frequency
  - Why is restarting good?
  - Efficient replay trick for frequent restart
- Which variable to split on
- Which branch to explore first
- Which lemmas to learn
- Blocked clause elimination
- Cache binary propagations
  - This is just scratching the surface

DPLL(*T*) solver interaction

T- Propagate $M \mid F, C \vee \ell \implies M, \ell^{C \vee \ell} \mid F, C \vee \ell$ $\quad C\ is\ false\ under\ T + M$

T- Conflict $M \mid F \implies M \mid F \mid \neg M'$ $\quad M' \subseteq M\ and\ M' is\ false\ under\ T$

T- Propagate $a > b, b > c \mid F, a \leq c \vee b \leq d \implies$

$$a > b, b > c, b \leq d^{a \leq c \vee b \leq d} \mid F, a \leq c \vee b \leq d$$

T- Conflict $M \mid F \implies M \mid F,\ a \leq b \vee b \leq c \vee c < a$

$$where\ a > b, b > c, a \leq c \subseteq M$$

Model based Theory Combination

Challenge:

- Solvers need to exchange what is equal.
- Computing all implied equalities is expensive.

Idea:

- Have solvers produce models.
- Use models to introduce equalities on demand.

If then guess

Summary

1. Progress in automated reasoning

   SAT, Automated Theorem Proving, SMT

1. An abstract account for SMT search (DPLL+T)

2. Integrating Theories

Takeaway: Theorem Proving is cool and beautiful