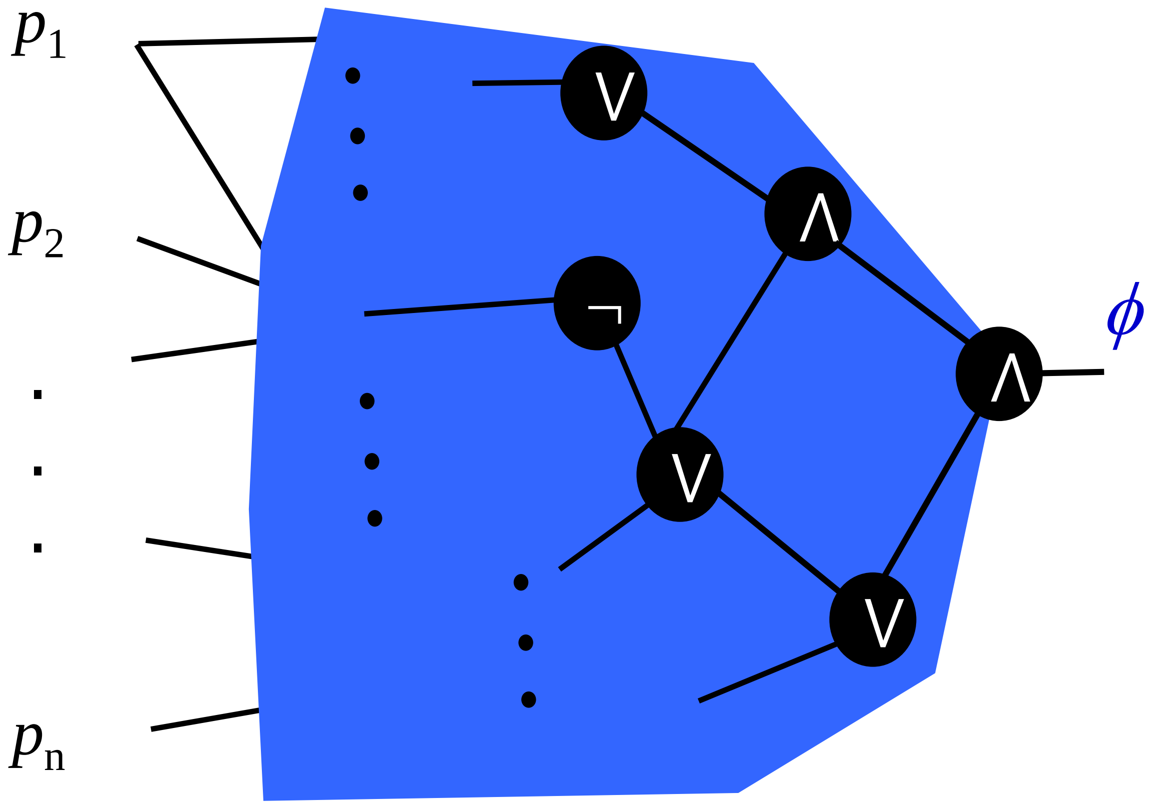
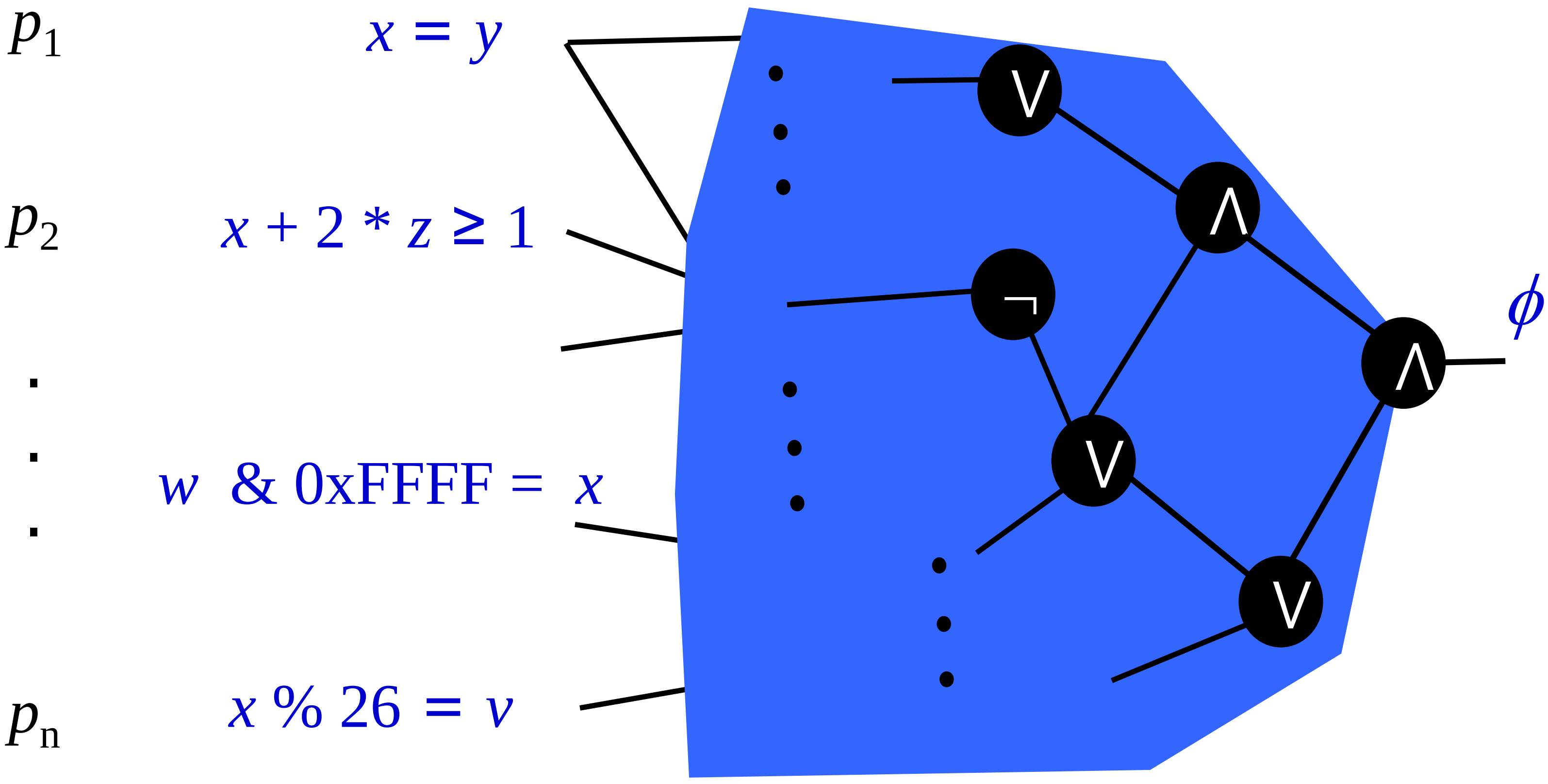
Boolean Satisfiability (SAT)



Is there an assignment to the $p_1, p_2, \ldots, p_n$ variables such that ϕ evaluates to 1?

Satisfiability Modulo Theories



Is there an assignment to the x,y,z,w variables s.t. ϕ evaluates to 1?

Motivation

- We have seen that efficient SAT solvers exit
  - DPLL is the most successful complete solver
- Can we generalize the results?
  - Is "$p \vee \neg q \vee (a = f(b - c)) \vee (g(g(b)) \neq c) \vee a - c \leq 7$" satisfiable?
- Improve our understanding of DPLL

From Propositional to First Order Logic

- $F ::= \exists X.\ F \mid \forall X.\ F \mid F \vee F \mid \neg F \mid r(\underline{t})$
- $t ::= f(\underline{t}) \mid c$
- Examples:
  - $\forall X.\ vote(X, trump) \Rightarrow \exists\ Y.\ vote(Y, klinton) \wedge Y = parent(X)$
  - $\forall X.\ \exists\ Y.\ Y * Y = X$

Satisfiability Modulo Theories

- Given a formula in first-order logic, with associated background theories, is the formula satisfiable?
  - Yes: return a satisfying solution
  - No [generate a proof of unsatisfiability]

Satisfiability Modulo Theories

- Any SAT solver can be used to decide the satisfiability of ground first-order formulas
- Often, however, one is interested in the satisfiability of certain ground formulas in a given first-order theory:
  - Pipelined microprocessors: theory of equality, atoms
    - $f(g(a, b), c) = g(c, a)$
  - Timed automata: planning: theory of integers/reals,
  - atoms
    - $x - y < 2$
  - Software verification: combination of theories, atoms
    - $5 + car(a + 2) = cdr(a[j] + 1)$
- We refer to this general problems as (ground) Satisfiability Modulo Theories, or SMT

Example Difference constraints

- Boolean combinations of `$a \leq b + k$'
  - a and b are free constants
  - $k \in Z$

Uninterpreted Functions

$\text{read}(\text{write}(X, Y, Z), Y) = Z$

$W \neq Y \Rightarrow \text{read}(\text{write}(X, Y, Z), W) = \text{read}(X, W)$

$x+2 = y \Rightarrow f(\mathit{read}(\mathit{write}(a, x, 3), y-2)) = f(y-x+1)$

A Simple Example(BMC)

Program

```
int x;
int y=8,z=0,w=0;
if (x)
  z = y - 1;
else
  w = y + 1;
assert (z == 5 ||
        w == 9)
```

constraints

y = 8,
z = x ? y – 1 : 0,
w = x ? 0 :y + 1,
z != 5,
w != 9

SMT counterexample found!

y = 8, x = 1, w = 0, z = 7

Motivating Example
Skolem-Lowenheim Formulas

- Prenex Normal Form $\exists\forall$
- $\exists x, y\ \forall z, w : P(x, y) \wedge \neg P(z, w)$

Lifting SAT to SMT

- Eager approach [UCLID]:
  - translate into an equisatisfiable propositional formula,
  - feed it to any SAT solver
- Lazy approach [CVC, ICS, MathSAT, Verifun, Zap]:
  - abstract the input formula into a propositional one
  - feed it to a DPLL-based SAT solver
  - use a theory decision procedure to refine the formula
- DPLL(T) [DPLLT, Z3, Sammy]:
  - use the decision procedure to guide the search of a DPLL solver

(Very) Lazy Approach for SMT – Example

$$g(a) = c \wedge \quad f(g(a)) \neq f(c) \vee g(a) = d \wedge c \neq d$$

$$1 \qquad \neg 2 \qquad 3 \qquad \neg 4$$

Send $\{1, \neg 2 \ \vee\ 3, \neg 4\}$ to the SAT solver

SAT solver returns $\{1, \neg\ 2, \neg 4\}$

Theory solver finds that $\{1, \neg 2\}$ is E-unsatisfiable

Send $\{1, \neg 2 \ \vee\ 3, \neg 4, \neg 1 \vee 2\}$ to the SAT solver

SAT solver returns $\{1, 2, 3\ , \neg 4\}$

Theory solver finds that $\{1, 3, \neg 4\}$ is E-unsatisfiable

Send $\{1, \neg 2 \ \vee\ 3, \neg 4, \neg 1 \vee 2, \neg 1 \vee \neg 3 \vee 4\}$ to the SAT solver

Return UNSAT

Decision Procedures

- Complete (terminating) algorithms for determining the validity (satisfiability) of a formula in a given logic
  - Cost is an issue
- Decidable logic a logic with a decision procedure for every formula
- Decidable (computation problem) there exists an a terminating algorithm which solves every instance of the problem

Obtaining a decision procedure

- Limit the logic
- Limit the class of intended models
- Answer validity (satisfiabilty) w.r.t. a given theory
  $T \models F$

Proving Decidability

- Small model theorem
  - Every satisfiable formula has a model whose size if proportional to the size of the formula
- Direct decision procedure
- Reduction to another decidable logic

Quantifier Free First Order Logic

- Universal formulas only
- Allow a fixed scheme of first order formulas T
- Determine if
  $T \models F$
- Decidable for interesting theories
  - Uninterpreted functions
    - $\forall a, b: f(a, b) = a \Rightarrow f(f(a, b), b) = a$
  - Theory of lists
  - Arrays
- Different theories can be combined

Theory of Uninterpreted Functions (EUF)

- Theory T $\forall X, Y: X = Y \Rightarrow f(X) = f(Y)$
- Determine the validity of universal formulas
- Decidability Ackerman 1954
- Downey, Sethi, Tarjan, Kozen, Nelson& Openn Efficient Algorithms
- Bryant, German, Velev Improvements for positive terms

Small model property of EUF formulas

- Ackerman 1954
- Every satisfiable formula has a model of size k where k is the number of distinct function application terms
- Example
  - $x = y \vee f(g(x)) = f\,(g(y))$
  - $\{x, y, g(x), g(y), f(g(x)), f(g(y))\}$
- Impractical algorithm

Proof by Refutation

- Determine the validity of a formula by checking the satifiability of its negation
- For quantifier free it is enough to consider Conjunction of literals
- Example "$\forall$A, B: f(A, B) = A$\Rightarrow$f(f(A, B), B) =A"
  - Proof that
    "f(a, b) = a $\wedge$ $\neg$ f((f(a, b), b) = a" is not satisfiable

An efficient EUF algorithm (intuition)

- Goal prove satisfiability of
  $t_1=u_1 \wedge \ldots \wedge t_p=u_p \wedge r_1 \neq s_1 \wedge \ldots \wedge r_q \neq s_q$
- Represent terms using DAGs
- Unify equal terms and their consequences
- Report UNSAT when contradicts inequalities
- Otherwise report SAT

The Congruent Closure Problem

- Given
  - A finite labeled directed graph G
    - Nodes are labeled by function symbols
    - Edges are labeled
  - A binary relation R on the nodes
- Two nodes are congruent under R if
  - They have the same label
  - Their arguments (outgoing neigbours) are in R (respectively)
- R is closed under congruences if all congruent nodes according to R are in R
- Compute the a minimal extension of R which is an equivalence relation and closed under congruences

Example 1

$$f(a, b) = a \land f((f(a, b), b) \neq a$$

Example 2

$$f(f(f(A))) = A \wedge f(f(f(f(f(A)))))=A \Rightarrow f(A) = A$$

$$f(f(f(a))) = a \wedge f(f(f(f(f(a)))))=a \wedge f(a) \neq a$$

Computing Congruence Closure

- Let R be a relation which is congruence closed
- Compute the congruence closure of $R \cup \{(u, v)\}$ by MERGE(u, v)

MERGE(u, v)

1. If FIND(u) = FIND(v) then return

2. Let P_u be the predecessors of vertices equivalent to u and P_v be the predecessors of vertices equivalent to to v

3. UNION(u, v)

4. For each pair (x, y) such that $x \in P_u$, $y \in P_v$, CONGRUENT(x, y) and FIND(x) $\neq$ FIND(y) do MERGE(x, y)

CONGRUENT(u, v) = label(u) = label(v) $\wedge \forall$i: FIND(u[i]) = FIND(v[i])

Properties of the Congruence Closure Algorithm

- Partial Correctness
- Complexity $O(m^2)$
- Downey, Sethi, and Tarjan achieves $O(m \log n)$ by storing the vertices in a hash table keyed by the list of equivalence classes of their successors

Application 1: EUF

- construct a graph G which corresponds to the set of all terms appearing in the conjunction $t_1=u_1 \wedge \ldots \wedge t_p=u_p \wedge r_1 \neq s_1 \wedge \ldots \wedge r_q \neq s_q$
- For each term i appearing in the conjunction let $\tau(i)$ denote the node of the term
- Let R be the identity relation on vertices
- For every $1 \leq i \leq p$, MERGE($\tau(t_i)$, $\tau(u_i)$)
- If for some $1 \leq j \leq q$, $\tau(r_j)$ is equivalent to $\tau(s_j)$) report UNSAT
- Otherwise report SAT

Improvements and Extensions

- Lahiri, Bryant, Goel, Talupur TACAS 2004
- Explicit Representation
  ITE(e1, e2, e3) = (e1 $\wedge$e2) $\vee$($\neg$e1$\wedge$e2)
  P(T1, T2, …, Tk)
- Treat `positive’ terms differently

Simple Theory of Lisp Lists

- car, cdr, cons without nil values
- Theory (axioms):

$$car(cons(X, Y)) = X$$

$$cdr(cons(X, Y)) = Y$$

$$\neg\, atom(X) \Rightarrow cons(car(X), cdr(X)) = X$$

$$\neg atom(cons(X,Y))$$

- Goal:

$$car(X)=car(Y) \wedge cdr(X) = cdr(Y) \wedge \neg atom(X) \wedge \neg atom(Y) \Rightarrow f(X) = f(Y)$$

- Use congruence closure with special equalities

Application 2: Lisp

- $v_1=w_1 \wedge \ldots \wedge b_r=w_r \wedge x_1 \neq y_1 \wedge \ldots \wedge x_s \neq y_s \wedge$ atom(u_1) $\wedge \ldots \wedge$atom(u_q)
- Construct a graph G which corresponds to the set of all terms appearing in the conjunction
- For each term i appearing in the conjunction let τ(i) denote the node of the term
- Let R be the identity relation on vertices
- For every $1 \leq i \leq r$, MERGE($\tau(v_i)$, $\tau(w_i)$)
- For every vertex u labeled by cons add a vertex v labeled by car and a vertex w labeled by cdr with out degree one s.t. v[1]=w[1]=u and MERGE(v, u[1]) and MERGE(v, u[2])
- If for some $1 \leq j \leq s$, $\tau(x_j)$ is equivalent to $\tau(y_j)$) report UNSAT
- If for some $1 \leq j \leq q$, $\tau(u_j)$ is equivalent to a cons node report UNSAT
- Otherwise report SAT

Integrating Values

$car(cons(X,Y)) = X$

$cdr(cons(X,Y)) = Y$

$X \neq nil \Rightarrow cons(car(X), cdr(X)) = X$

$cons(X,Y) \neq nil$

$car(nil) = cdr(nil) = nil$

- Becomes NP-Hard

Theory of Arrays (Stores)

- read(write(v, i, e), j) =
  if i=j then e else read(v, j)
- write(v, i, read(v, i)) = v
- write(write(v, i, e), i, f) = write(v, i, f)
- i $\neq$j $\Rightarrow$ write (write (v, i, e), j, f) =
  write (write (v, j, f), I, e)
- Eliminate write and use EUF

Combining Decision Procedures

- Programming languages combine different features
  - Arithmetic
  - Data types
  - Arrays
  - …
- Is there a way to compose decision procedures of different theories?
- Given two decidable logics is there a way to combine the logics into a decidable logic?

Bibliography

- Nelson & Oppen
  Fast Decision Procedures Based on Congruence Closure
  JACM 1979
- Stump, Dill, Barrett, Levitt
  A Decision Procedure for an Extensional Theory of Arrays
  LICS'01

Combining Decision Procedures

- Programming languages combine different features
  - Arithmetic
  - Data types
  - Arrays
  - …
- Is there a way to compose decision procedures of different theories?
- Given two decidable logics is there a way to combine the logics into a decidable logic?

Cooperating Decision Procedures Nelson & Oppen

- Quantifier free
- Proof be refutation
- Separate the conjunct into separate conjuncts
  $A \wedge B$
  such that
  - A and B use different theories
  - Only constants are shared
- If either A or B is UNSAT report UNSAT
- When A and B are SAT propagate equalities between A and B and repeat

Example Theories

LIST

$car(cons(X, Y)) = X$

$cdr(cons(X, Y)) = Y$

$\neg atom(X) \Rightarrow cons(car(X), cdr(X)) = X$

$\neg atom(cons(X, Y))$

EUF

$X = Y \Rightarrow f(X) = f(Y)$

R

$X+0 = 0$

$X + -X = 0$ $\quad$ $0 \neq 1$

$(X+Y)+Z = X + (Y+Z)$ $\quad$ $0 \leq 1$

$X+Y = Y + X$

$X \leq X$

$X \leq Y \vee Y \leq X$

$X \leq Y \wedge Y \leq X \Rightarrow X = Y$

$X \leq Y \wedge Y \leq Z \Rightarrow X \leq Z$

$X \leq Y \Rightarrow X+Z \leq Y+Z$

A Simple Example

$$x \leq y \wedge y \leq x + \overset{g_1}{car}(cons(\overset{g_5}{0}, x)) \wedge P(\overset{g_2}{\underset{g_3}{h(x)}-\underset{g_4}{h(y)}}) \wedge \neg P(\underset{g_5}{0})$$

$x \leq y$
$y \leq x + g_1$
$g_2 = g_3 - g_4$
$g_5 = 0$

$P(g_2)=true$
$P(g_5)=false$
$g_3=h(x)$
$g_4=h(y)$

$g_1 = car(cons(g_5, x))$

Equality Propagation Procedure

1. Assign conjunctions to F_L and F_F s.t.,
   - F_F contains only F-literals
   - F_L contains only L-literals
   - $F_L \wedge F_F$ is satisfiable iff F is satisfiable
2. If either F_L or F_F is UNSAT report UNSAT
3. If either F_L or F_F entails equality not entailed by other add this equality and go to step 2
4. If either F_L or F_F entails $u_1=v_2 \vee u_2=v_2 \vee \ldots u_k=v_k$ without entailing any equality alone then apply the procedure recursively to the k-formulas
   $F_L \wedge F_F \wedge v_i = u_i$
   If any of these formulas is SAT return SAT
5. Return UNSAT

Notes

- Only equalities are propagated
- Requires that the theories can find all consequent equalities
- Completeness is non-obvious
- The original paper also performs simplification

Convexity

- A formula F is non-convex F entails
  $u_1=v_2 \vee u_2=v_2 \vee \ldots\ u_k=v_k$
  without entailing any equality alone
  - Otherwise it is convex
- A theory is convex
- Convex theories
  - EUF
  - Relational linear algebra
- Non-convex theories
  - Theory of arrays
  - Theory of reals under multiplications
    $xy=0 \wedge z=0 \models_R x=z \vee y=z$
  - Theory of integers under + and $\leq$

Hints about Completeness

- The residues of formula
  - The strongest Boolean combinations of equalities between constants entailed by the formula

$x=f(a) \wedge y=f(b)$	$a=b \rightarrow x=y$
$x+y-a-b>0$	$\neg(x=a \wedge y=b) \wedge \neg(x=b \wedge y=a)$
$x=write(v, u, e)[j]$	$i=j \rightarrow x=e$
$x=write(v, u, e)[j] \wedge$ $y=v[j]$	if $i=j$ then $x=e$ else $x=y$

~~Lemma 4: If A and B are formulas whose only common parameters are constant symbols then RES($A \wedge B$) = RES(A)$\wedge$RES(B)~~

More correct account of completeness

- A theory T is stably infinite if every quantifier-free formula is T-satisable if and only if it is satisfied by a T-model A whose domain A is infinite
- For lemma 4 we require
  - The theories are disjoint
  - Both theories are stably infinite
  - Read more in Manna 2003

The residues in the simple example

$x \leq y \wedge y \leq x + car(cons(0, x)) \wedge P(h(x)-h(y)) \wedge \neg P(0)$

$x \leq y$
$y \leq x + g_1$
$g_2 = g_3 - g_4$
$g_5 = 0$
$g_1 = g_5 \rightarrow x = y \wedge$ $g_5 = g_2 \leftrightarrow g_3 = g_4$

$P(g_2) = true$
$P(g_5) = false$
$g_3 = h(x)$
$g_4 = h(y)$
$g_2 \neq g_5 \wedge$ $x = y \rightarrow g_3 = g_4$

$g_1 = car(cons(g_5, x))$
$g_1 = g_5$

Handling Quantifiers

- The problem becomes undecidable
- Refutationally resolution based complete procedures exist and implemented (e.g., SPASS, Vampiere)
  - Not guaranteed to terminate
  - Do not handle theories
- Z3 employs incomplete heuristics
  - Instantiate universal quantifiers with relevant terms
  - Can be tuned by the user

Conclusion

- Handling specialized theories yields significant improvements
  - Efficiency
  - Termination
  - Predictability
- Combination procedures are useful
- But resolution based theorem provers can still br superior in several cases

Partial Bibliography

- Nelson & Oppen
  Simplification by Cooperating Decision Procedures
  TOPLAS 1980
- Tinelli & Zarba
  Combining non-stably infinite theories
  Journal of Automated Reasoning, 2006
- **Simplify: A Theorem Prover for Program Checking**
  David Detlefs, Greg Nelson, James B. Saxe
  JACM 2005
- **Combining Decision Procedures (2003)**
  Zohar Manna, Calogero G. Zarba
- **Verifun: An Explicating Theorem Prover for Quantified Formulas**
  Cormac Flanagan, Rajeev Joshi, James B. Saxe
  HPL-2004-199
- Leonardo Mendonça de Moura, Nikolaj Bjørner:
  Satisfiability modulo theories: introduction and applications. Commun. ACM 54(9): 69-77 (2011)
- Leonardo Mendonça de Moura, Nikolaj Bjørner:
  Engineering DPLL(T) + Saturation. IJCAR 2008: 475-490