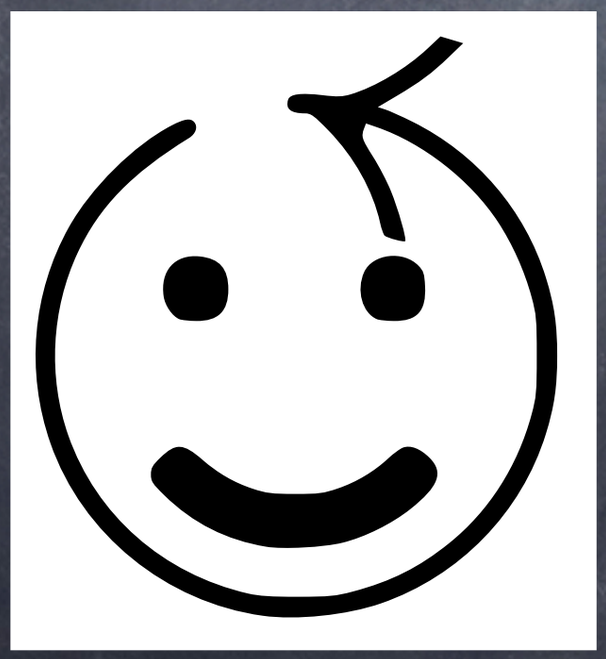
Introduction to Max-SAT and Max-SAT evaluation

(slightly revised version)



Masahiro Sakai

2014-02-27 @ ZIB Berlin

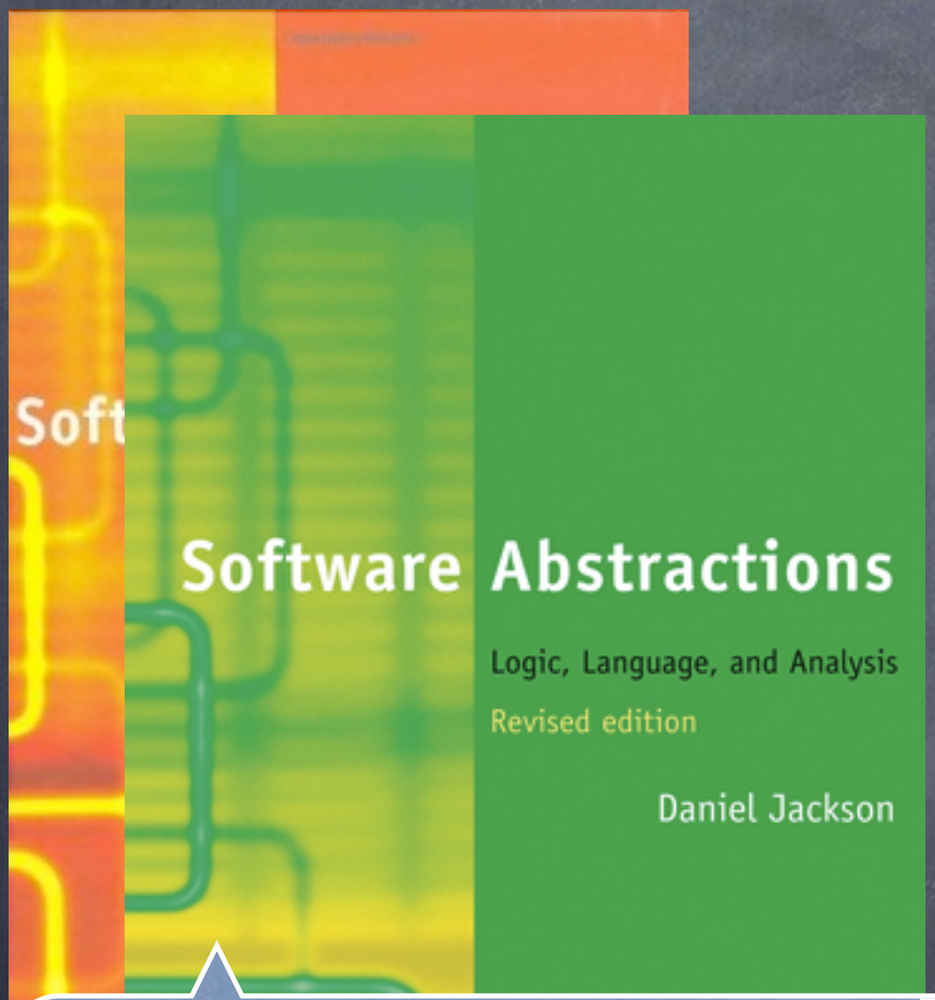
Today's topics

- About Myself
- SAT and related problem classes
- My experience of Max-SAT evaluation 2013
- Towards Max-SAT Evaluation 2014
- Conclusion

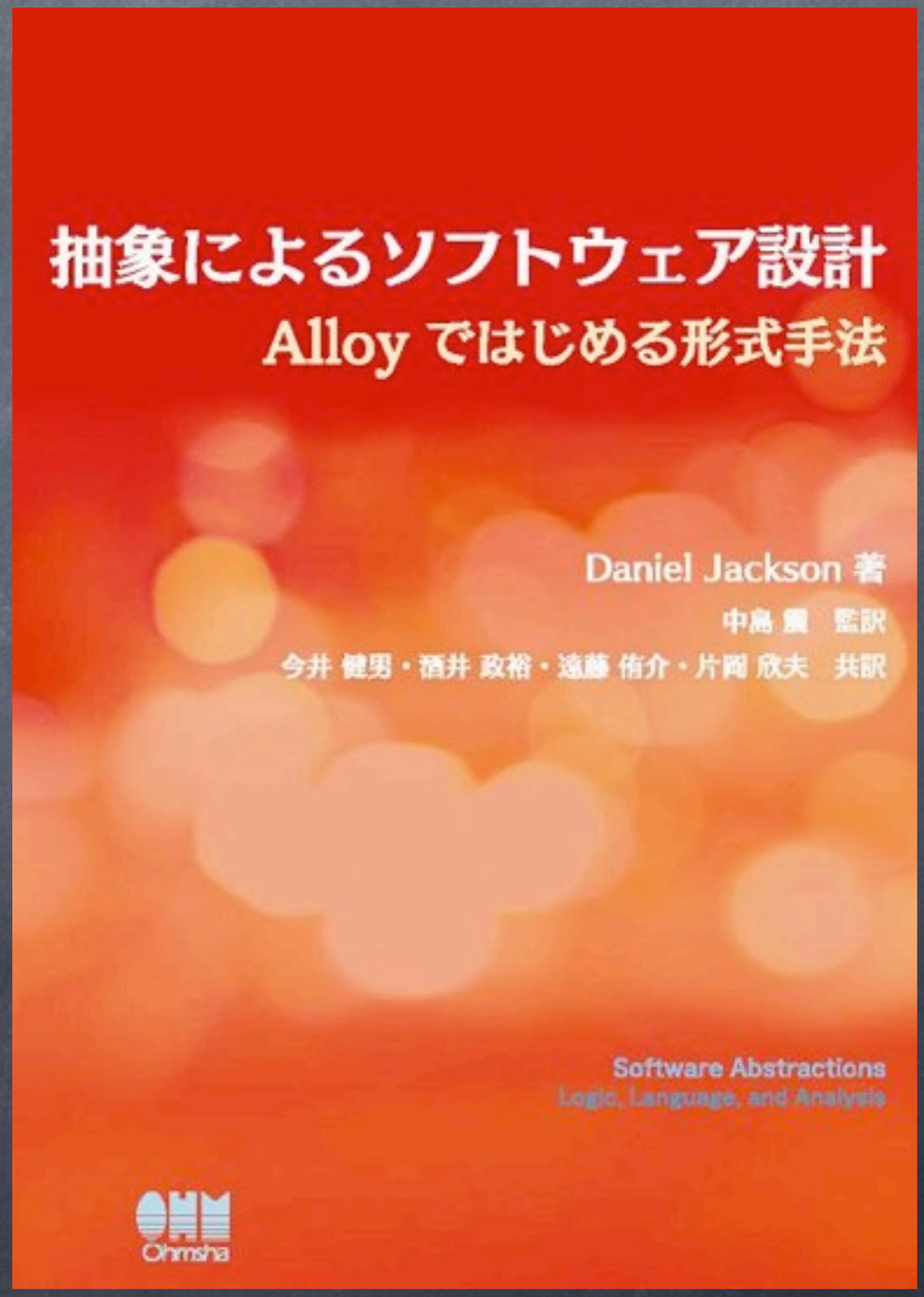
About myself

- Masahiro Sakai (酒井 政裕), from Japan
- I'm NOT an expert of "MATHEMATICAL PROGRAMMING" nor "OPERATIONS RESEARCH"
- My background
  - Logic, Programming Language Theory (Domain Theory, Type Theory, Functional Programming), Category Theory, etc.
- My job at TOSHIBA
  - Software Engineering
    - Model Checking, Specification Mining, etc.
  - Recommendation System

"Software Abstractions"

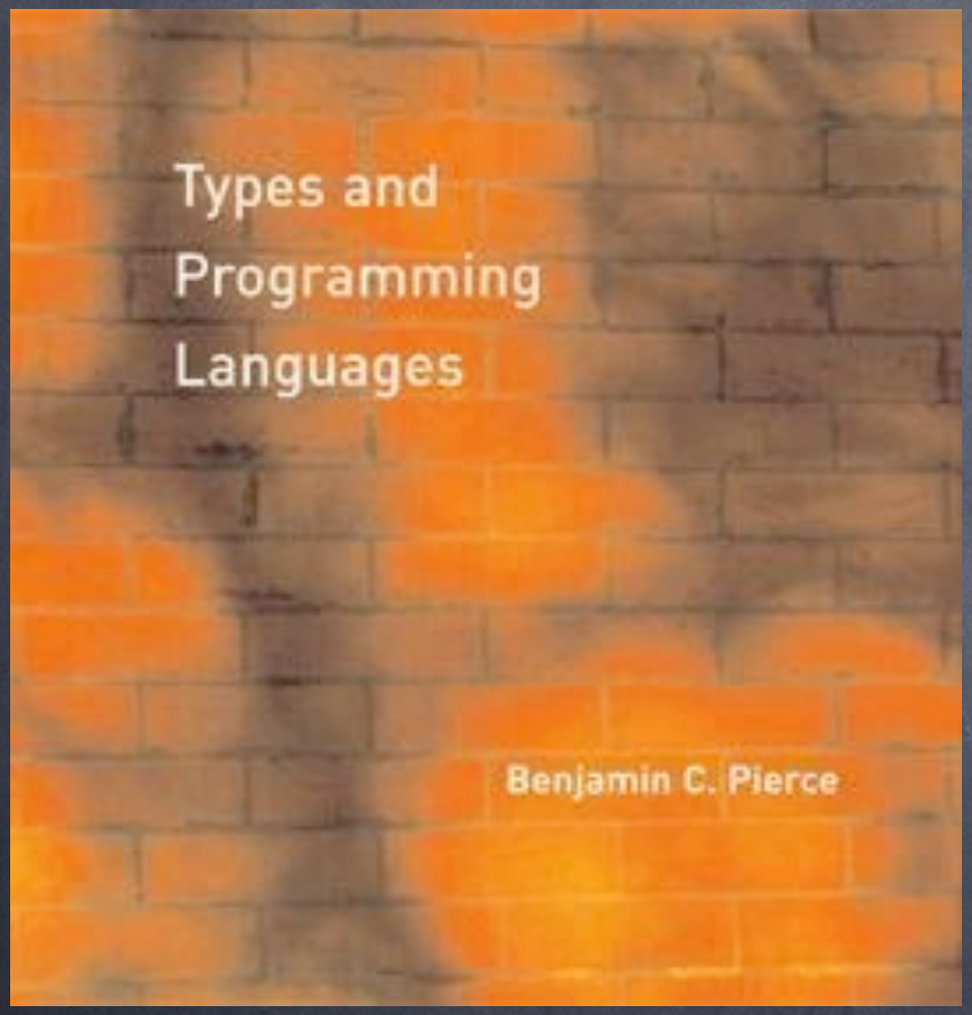


Japanese translation



Textbook about "Formal Methods" and Alloy tool

"Types and Programming Languages"



Japanese translation

Textbook about "type systems"

About myself (cont'd)

- My recent interests:
  - **Decision procedures** or **Solver algorithms**
- Because of
  - my interests in logic
    - e.g. I'm impressed by decidability of Presburger arithmetic and theory of real closed fields.
  - prevalent usage of SAT/SMT solvers in software engineering (Alloy is one example)
- Along the way, I also got interested in mathematical programming.
- I implemented several toy-level implementations of these algorithms.

My hobby project "toysolver"/"toysat": toy-level implementations of various algorithms

- Integer Arithmetic
  - Cooper's Algorithm
  - Omega Test
  - Gomory's Cut
  - Conti-Traverso
  - Branch-and-bound
- Real Arithmetic
  - Fourier-Motzkin variable elimination
  - Gröbner basis (Buchberger algorithm)
  - Cylindrical Algebraic Decomposition
  - Simplex method
- Uninterpreted functions
  - Congruence Closure
- SAT / MaxSAT / Pseudo Boolean

github.com/msakai/toysolver

SAT and related problems

SAT

- SAT = SATisfiability problem (of propositional logic)
  - "Given a propositional formula φ containing propositional variables,
    Is there a truth assignment M that makes the formula true? (i.e. $M \models \varphi$)"
- SAT solver decides the SAT problem
  - When the formula is satisfiable,
    it also produce one such truth assignment.

SAT: Examples

- Example 1: $(P \vee Q) \wedge (P \vee \neg Q) \wedge (\neg P \vee \neg Q)$
  - → "Satisfiable" (or "Feasible" in mathematical programming term)
  - Assignments: (P,Q) = (TRUE, FALSE))
- Example 2: $(P \vee Q) \wedge (P \vee \neg Q) \wedge (\neg P \vee \neg Q) \wedge (\neg P \vee Q)$
  - → "Unsatisfiable" (or "Infeasible" in mathematical programming term)

- Note:
  - Input formula is usually given in CNF (conjunction normal form)
    - CNF ::= Clause $\wedge$ ... $\wedge$ Clause
    - Clause ::= Literal $\vee$ ... $\vee$ Literal
    - Literal ::= Variable | $\neg$Variable
  - Non-CNF formula can be converted to equi-satisfiable CNF in linear size by introducing auxiliary variables. ("Tseitin encoding", similar to linearization of 0-1 integer programing)

Why SAT solvers attract attentions?

- SAT is a classical and canonical NP-complete problem.
- But SAT solvers speed up drastically in last 15 years
  - State-of-art SAT solver can solve problems of millions of variables and constraints.
- Many applications in software engineering and other fields, now encode their problems into SAT/SMT and use off-the-shelf SAT/SMT solver.
  - to utilize the advances of off-the-shelf solvers
  - to separate two concerns:
    - problem formulation which requires domain knowledge
    - solving algorithms

SAT: Basic algorithm

- Classical algorithm: DPLL (Davis-Putnam-Logemann-Loveland) algorithm
  - Tree-search algorithm
  - Constraint propagation called unit propagation
    - All except one literals in a clause become false, the remaining literal is assigned to true.
- Modern improvements
  - CDCL (Conflict-driven clause learning)
  - Non-chronological backtracking
  - Efficient data-structure for constraint propagation
  - Adaptive variable selection heuristics
  - Restarts, Conflict Clause Minimization, Component caching, etc.

Related problems: Max-SAT and Pseudo Boolean Satisfaction/Optimization

Related Problems

"Computer Algebra" field

More Arithmetical

RCF (Real Closed Field)

more like optimization

"Mathematical Programming" field

PBS

PBO

Integer Programming

SMT

Automatic Theorem Proving

Finite Model Finding

SAT

Max SAT

QBF

Focus of this talk.

"Theorem Proving" field

propositional logic to first-order logic

This is a rough overview.
Ignore detailed positions

Max-SAT

- Max-SAT is an optimization extension of SAT
  - "Given a set of clauses, find an assignment that maximize satisfied clause."
- Unlike its name, it's common to formulate as minimization of VIOLATED clauses.
- Example:
  - $\{\neg P_1 \vee \neg P_2,\ \neg P_1 \vee P_3,\ \neg P_1 \vee \neg P_3,\ \neg P_2 \vee P_4,\ \neg P_2 \vee \neg P_4,\ P_1,\ P_2\}$
  - $\rightarrow (P_1, P_2, P_3, P_4)=(F, F, F, T)$, 2 clauses are violated

Partial / Weighted variants of Max-SAT

- Partial Max-SAT:
  - HARD and SOFT clauses
- Weighted Max-SAT:
  - each clause has associated cost
  - minimize the total costs of violated clauses
- Weighted Partial Max-SAT:
  - obvious combination of the two

Example of
Weighted Partial Max-SAT

$x1 \vee \neg x2 \vee x4$

$\neg x1 \vee \neg x2 \vee x3$

[8] $\neg x2 \vee \neg x4$

[4] $\neg x3 \vee x2$

[3] $x1 \vee x3$

Some Algorithms to solve Max-SAT family

- Convert to Pseudo Boolean Optimization (PBO) or Integer Programming problems.
- Branch-and-Bound
  - w/ modified version of unit-propagation
  - w/ specific lower bound computation (e.g. using disjoint inconsistent subsets)
- Unsatisfiability-based (or core-guided) approach
- Hybrids of those

Conversion to PBO (1)

- Some SAT solvers are extended to handle more expressive constraints than clauses
  - Clause
    - "$L_1 \vee \ldots \vee L_n$" $\Leftrightarrow$ "$L_1 + \ldots + L_n \geq 1$"
      if truth is identified with 1-0
  - Cardinality constraints
    - "at least k of $\{L_1, \ldots, L_n\}$ is true" $\Leftrightarrow$ "$L_1 + \ldots + L_n \geq k$"
  - Pseudo boolean constraints
    - integer-coefficient polynomial inequality constraints over literals
    - e.g. $2\ L_1 + 2\ L_2L_3 + L_4 \geq 3$

Conversion to PBO (2)

- Pseudo boolean satisfaction (PBS)
  - satisfiability problems of pseudo-boolean constraints
- Pseudo boolean optimization (PBO)
  - PBS with objective function
  - ≅ (non-linear) 0-1 integer programming

Conversion to PBO (3)

$x1 \vee \neg x2 \vee x4$

$\neg x1 \vee \neg x2 \vee x3$

[8] $\neg x2 \vee \neg x4$

[4] $\neg x3 \vee x2$

[3] $x1 \vee x3$

[2] $x6$

Minimize

$8\ r3 + 4\ r4 + 3\ r5 + 2\ \neg x6$

Subject to

$x1 + \neg x2 + x4 \geq 1$

$\neg x1 + \neg x2 + x3 \geq 1$

$r3 + \neg x2 + \neg x4 \geq 1$

$r4 + \neg x3 + x2 \geq 1$

$r5 + x1 + x3 \geq 1$

- r_is are relaxation variables for Soft clause.
- Unit clause (e.g. x6) does not need a relaxation variable.
- Further conversion to 0-1 ILP is obvious.

PBS/PBO algorithm

- PBS
  - SAT solver is extended to handle pseudo-boolean constraints
    - Sometimes "cutting-plane proof system" instead of "resolution" is used for conflict analysis / learning.
- PBO
  - Satisfiability-based approach
  - Branch-and-Bound
  - Unsatisfiability-based approach

PBO: Satisfiability-based algorithm

Minimize $\Sigma^{n}_{j=1}\ c_j\ l_j$
Subject to P

```
M ← None
UB ← ∞
while true
   if P is SAT
     M ← getAssignments()
     UB ← Σ^n_{j=1} c_j M(l_j)
     P ← P ∧ (Σ^n_{j=1} c_j l_j < UB)
   else
     return M and UB
```

Many SAT solver allows incrementally adding constraints and re-solving. (It is faster than solving from scratch, since learnt lemma and other info are reused.)

This is linear search on objective values, but binary search and more sophisticated search are also used.

PBO: Branch-and-Bound

- Various lower-bound computation methods are used.
  - LP relaxation
  - MIS (Maximum Independent Set) lower bounding
  - Lagrange relaxation
- Note
  - LP relaxation is tighter, but more time-consuming.
  - Therefore, sometimes, more lax but cheeper methods are preferred.

Unsatisfiability-based Max-SAT algorithm

Treat all SOFT-clauses as HARD clauses, and invoke SAT solver.
If unsatisfiable, relax the unsatisfiable subset, and solve again.
First satisfiable result is the optimal solution.

$\varphi_W \leftarrow \varphi$
while (φ_W is UNSAT)
  do Let φ_C be an unsatisfiable sub-formula of φ_W
    $V_R \leftarrow \varnothing$
    for each soft clause $\omega \in \varphi_C$
      do $\omega_R \leftarrow \omega \cup \{ r \}$
        $\varphi_W \leftarrow (\varphi_W \setminus \{ \omega \}) \cup \{\omega_R \}$
        $V_R \leftarrow V_R \cup \{ r \}$
    $\varphi_R \leftarrow \{ \Sigma\_\{r \in V_R\} \; r \leq 1 \}$
    $\varphi_W \leftarrow \varphi_W \cup \varphi_R$ // Clauses in φ_R are declared hard
return $|\varphi|$ − number of relaxation variables assigned to 1

Can be extended for weighted version, but omitted here for simplicity.

For detail, see "Improving Unsatisfiability-based Algorithms for Boolean Optimization" by Vasco Manquinho, Ruben Martins and Inês Lynce.

Remark: Semidefinite Optimization Approaches

- SDP (Semi-definite programming) relaxation of Max-SAT is known to be tighter than LP relaxation.
- There are beautiful results on approximate algorithms based on SDP relaxation
- Still there are no practical Max-SAT solver that incorporate SDP, AFAIK.

Max-SAT Evaluation 2013

Max-SAT evaluation

- Max-SAT evaluation is the annual Max-SAT solver competition
  - one of the solver competitions affiliated with SAT-conference
- Why I submitted to Max-SAT 2013?
  - I submitted my "toysat" to the Pseudo Boolean Competition 2012 (PB12) one year ago, and its performance was not so bad in some categories, considering it was not tuned up well.
  - I wanted to re-challenge, but it was the last PB competition, so I moved to Max-SAT evaluation.

Results of PB12: PBS/PBO track

- DEC-SMALLINT-LIN
  - 1st: SAT 4j PB RES // CP, …, 19th: SCIP spx standard, 20th: toysat
- DEC-SMALLINT-NLC
  - 1st: pb_cplex, 2nd: SCIP spx standard, …, 18th: toysat, …
- DEC-BIGINT-LIN
  - 1st: minisatp, …, 4th: SCIPspx, 16th: toysat, …
- OPT-SMALLINT-LIN
  - 1st: pb_cplex, 2nd: SCIP spx E, …, 24th: toysat, …
- OPT-SMALLINT-NLC
  - 1st: SCIP spx E, …, 27th: toysat, …
- OPT-BIGINT-LIN
  - 1st: SAT4J PB RES // CP, …, 8th: toysat, …

DEC: decision problem (PBS)
OPT: optimization problem (PBO)
SMALLINT: all coefficients are $\leq 2^{20}$
BIGINT: some coefficients are $>2^{20}$
LIN: linear constraints/objective
NLC: non-linear …

Results of PB12: WBO track

- PARTIAL-BIGINT-LIN
  - 1st: Sat4j PB, 2nd: toysat, 3rd: wbo2sat, ...
- PARTIAL-SMALLINT-LIN
  - 1st: SCIP spx, 2nd: clasp, 3rd: Sat4j PB, 4th: toysat, ...
- SOFT-BIGINT-LIN
  - 1st: Sat4j PB, 2nd: toysat, 3rd: npSolver, ...
- SOFT-SMALLINT-LIN
  - 1st: SCIP spx, 2nd: Sat4j PB, 3rd: clasp, 4th: toysat, ...

My submission to Max-SAT 2013

- toysat: my own SAT-based solver
  - Simple incremental SAT-solving using linear search on objective values
    - (since other features are not tuned up / tested enough)
- scip-maxsat:
  - SCIP Optimization Suite 3.0.1 with default configuration
  - Simple conversion to 0-1 ILP.
- glpk-maxsat
  - GLPK 4.45 with default configuration.
  - Simple conversion to 0-1 ILP.

Results of Max-SAT 2013

	MaxSAT	Weighted	Partial	W. Partial
Random	MaxSatz2013f	ISAC+	ISAC+	Maxsatz2013f
	ISAC+	Maxsatz2013f	WMaxSatz+	ISAC+
	WMaxSatz09	ckmax–small	WMaxSatz09	WMaxSatz09
Crafted	ISAC+	ISAC+	ISAC+	MaxHS
	Maxsatz2013f	Maxsatz2013f	SCIP-maxsat	ISAC+
	WMaxSatz09	WMaxSatz+	ILP	ILP
Industrial	pMiFuMax	—	ISAC+	ISAC+
	ISAC+	—	QMaxSAT2-mt	WPM1-2013
	WPM1	—	MSUnCore	wMiFuMax

- This time, toysat did not perform well :(

My opinion: Benefits of submitting a solver to competitions

- You can benchmark your solver with others for FREE.
  - Benchmarking solvers is a hassle.
    - Requires lots of resources.
    - Not all solvers are open-source.
- There are various sets of benchmarks, which sometimes reveal subtle bugs in your solver.
  - PB12 revealed a subtle bug of toysat in conflict analysis of pseudo boolean constraints.
  - Max-SAT 2013 revealed two bugs in SCIP/SoPlex.

Max-SAT 2013: Buqs of SCIP

- Organizers notified me that SCIP-maxsat produced wrong results on some instances, and I resubmitted fixed version.
- Case 1:
  - Running out of memory in SoPlex-1.7.1 LP solver leads to declare non-optimal solution as optimal.
    - As an ad-hoc measure, I simply modified SoPlex to terminate its process immediately after memory allocation error w/o exception handling.
- Case 2:
  - SCIP produced wrong optimal result on ONLY one instance of the competition.
  - SCIP-2.1.1 worked correctly, but SCIP-3.0.1 did not!
  - Michael Winkler fixed the bug. Thanks!!

Towards Max-SAT Evaluation 2014

My Submission Plan

- Important dates
  - Submission deadline: April 11, 2014
  - Results of the evaluation: July 8-12, 2014
- My plan
  - SCIP and FibreSCIP
    - If you do not submit by yourselves, I'll submit instead.
    - I'd like to know a configuration that is better than default one.
  - toysat
    - I'm tuning its core implementation now, and I'll adopt more sophisticated algorithm than linear incremental SAT solving (e.g. core-guided binary search).

Concluding Remarks

Interaction between AI/CP and OR community

- Now the two fields are overlapping, and interactions between them are active/interesting research area.
- Examples:
  - SCIP incorporates techniques from SAT/CP.
  - Cutting-plane proof systems in SAT-based PB solvers.
  - SMT solvers use Simplex, cutting-plane, etc. as "theory solvers" for arithmetic theory.
- I hope this direction yields more fruitful results in the future.

Related Problems

"Computer Algebra" field

More Arithmetical

RCF (Real Closed Field)

more like optimization

"Mathematical Programming" field

PBS

PBO

Integer Programming

SMT

Automatic Theorem Proving

Finite Model Finding

SAT

Max SAT

QBF

Focus of this talk.

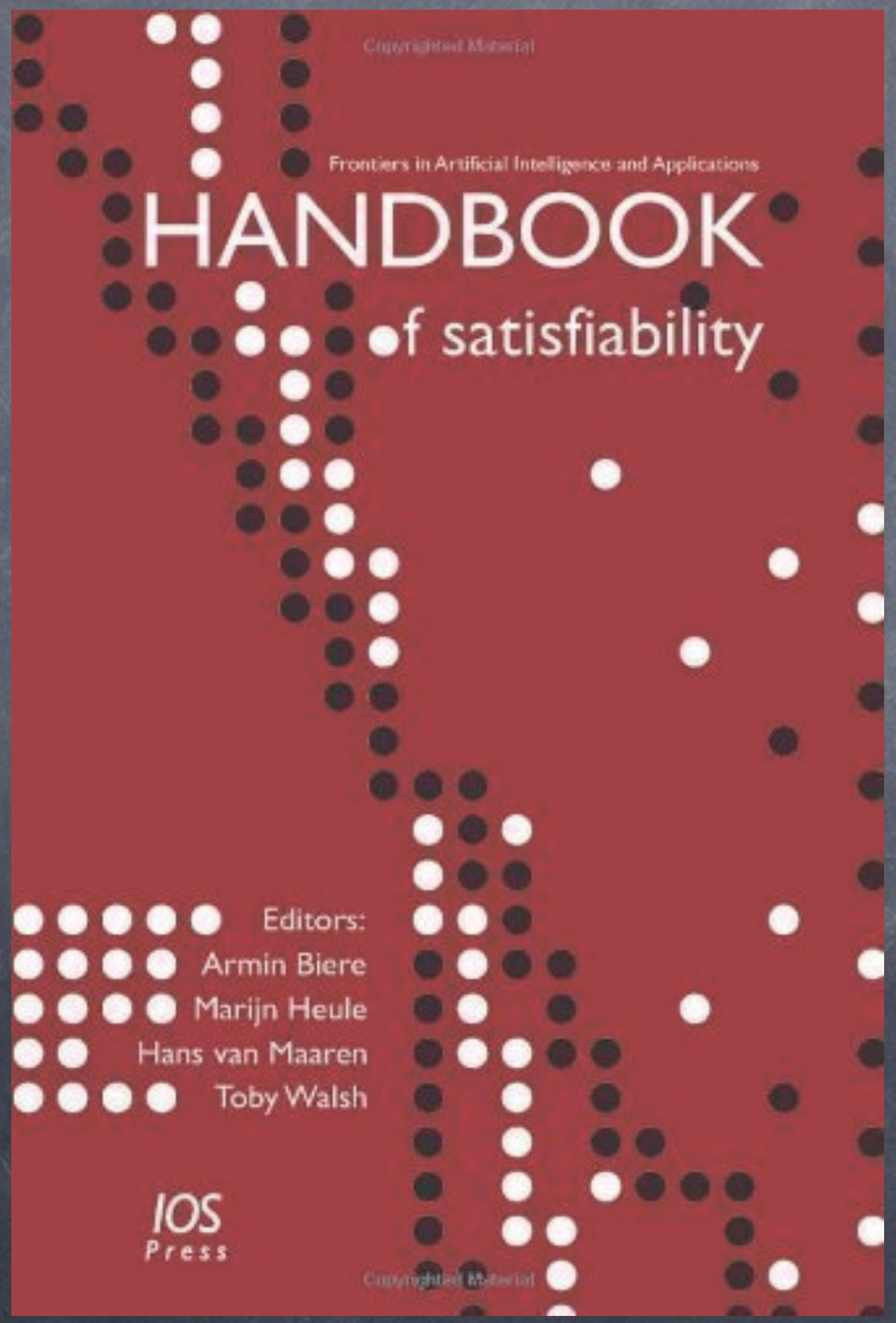
"Theorem Proving" field

propositional logic to first-order logic

This is a rough overview.
Ignore detailed positions

Any Questions?

Reference


Copyrighted Material
Frontiers in Artificial Intelligence and Applications
HANDBOOK
of satisfiability
Editors:
Armin Biere
Marijn Heule
Hans van Maaren
Toby Walsh
IOS
Press
Copyrighted Material

Reference

How a CDCL SAT Solver works

Masahiro Sakai
Twitter: @masahiro_sakai

12年9月23日日曜日

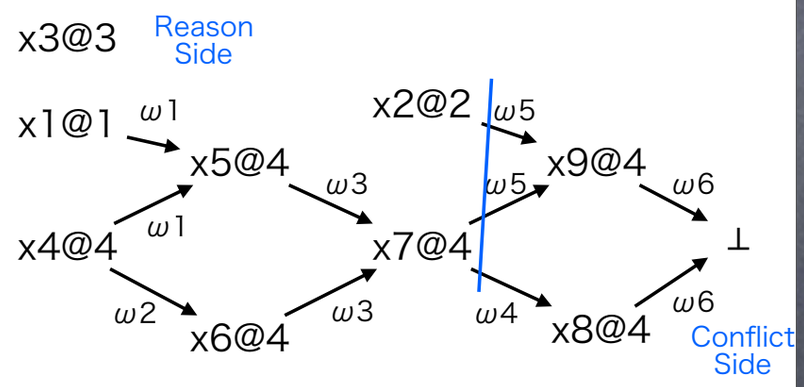
Diagnose

$\omega 6$: $\neg x8 \vee \neg x9$ $\omega 4$: $\neg x7 \vee x8$

$\neg x7 \vee \neg x9$ $\omega 5$: $\neg x2 \vee \neg x7 \vee x9$

$\neg x2 \vee \neg x7$

Implication Graph



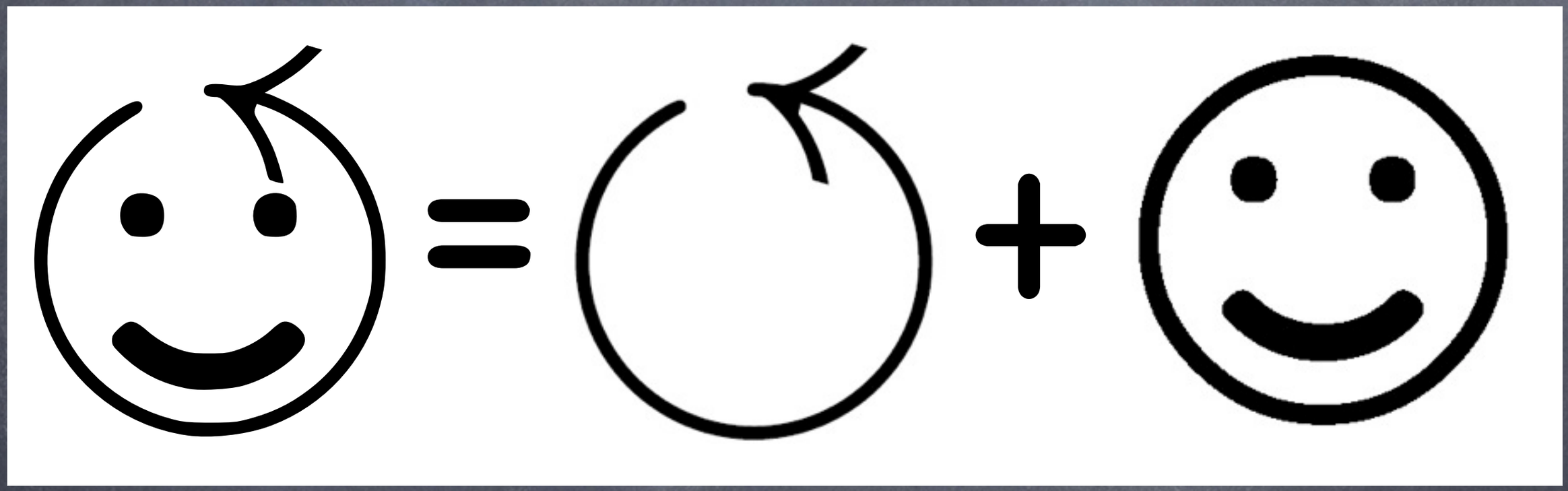
Clause DB

- $\omega 1$: $\neg x1 \vee \neg x4 \vee x5$
- $\omega 2$: $\neg x4 \vee x6$
- $\omega 3$: $\neg x5 \vee \neg x6 \vee x7$
- $\omega 4$: $\neg x7 \vee x8$
- $\omega 5$: $\neg x2 \vee \neg x7 \vee x9$
- $\omega 6$: $\neg x8 \vee \neg x9$
- $\omega 7$: $\neg x8 \vee x9$

12年9月23日日曜日

http://www.slideshare.net/sakai/how-a-cdcl-sat-solver-works

Appendix:
About my icon



Commutativity makes
category theorists happy!