Aalto University
School of Science
and Technology

Classifying and Propagating Parity Constraints

Tero Laitinen, Tommi Junttila, Ilkka Niemelä

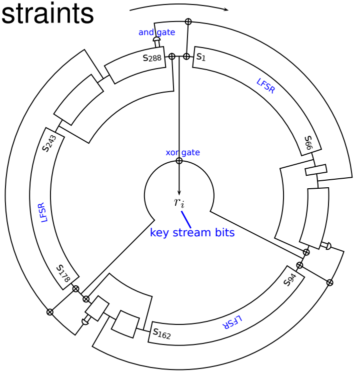
Department of Information and Computer Science
Aalto University, School of Science
{tero.laitinen,tommi.junttila,ilkka.niemela}@aalto.fi

October 12, 2012

Background: Parity Constraints

- xor-clause $l_1 \oplus \cdots \oplus l_n$: odd number of literals "true"
  - $\rightsquigarrow$ linear equation: $a_1 x_1 + \cdots + a_n x_n \equiv 1 \pmod 2$
- application domains with parity constraints
  - circuit verification
  - bounded model checking
  - logical cryptanalysis
- structure lost in CNF

$$x \oplus y \oplus z \rightsquigarrow \begin{cases} x \vee y \vee z \\ x \vee \neg y \vee \neg z \\ \neg x \vee y \vee \neg z \\ \neg x \vee \neg y \vee z \end{cases}$$



- Gaussian elimination
  - solves parity constraints in polynomial time
  - not applicable with nonlinear constraints (or-clauses)

Background: CNF-XOR SAT Problem

- **cnf-xor instance** : or-clauses $\wedge$ xor-clauses
- **cnf-xor SAT problem** : Given a cnf-xor instance, decide whether it is satisfiable.

Example

Instance: $(\neg x \vee y) \wedge (\neg y \vee \neg z) \wedge (x \oplus y \oplus z \oplus \top)$:

- solution $\{x, y, \neg z\}$

$\Rightarrow$ goal : effective SAT solver for cnf-xor SAT problem

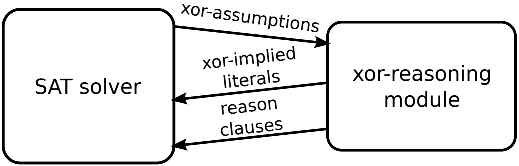
Background : satisfiability and parity constraints

- Modern clause learning SAT solvers
  - perform usually very well
  - but tend to scale poorly with parity constraints
- DPLL(XOR) framework, Laitinen et al. ECAI 2010
  - xor-reasoning SMT module
  - many propagation engines
  - .. but which one to use?
- **this work**:
  - fast approximating tests for detecting whether unit propagation or equivalence reasoning is “enough”
  - translations for propagating parity constraints faster, e.g. simulating equivalence reasoning with unit propagation

Outline

1. DPLL(XOR) framework
2. Classifying parity constraints
3. Simulating equivalence reasoning
4. Experimental results

DPLL(XOR) Framework



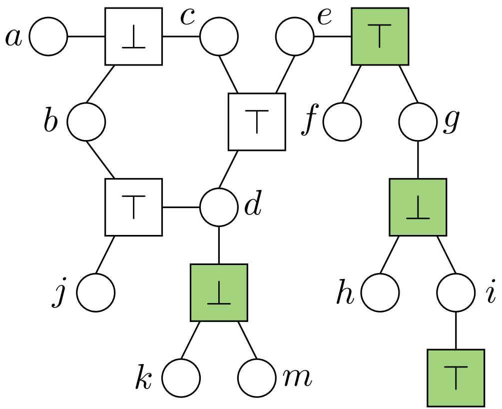
- SAT solver
  - conflict-driven clause learning search on cnf-part
- xor-reasoning module
  - DPLL(T)-style SMT module for SAT solver, variables shared
  - checks satisfiability of xor-part
  - infers truth values using xor-part
  - computes reason clauses
- related work
  - EqSatz, march_eq, MoRsat, CryptoMinisat, lsat

Classifying Parity Constraints

- compared to unit propagation, parity reasoning is computationally intensive
- fast structural approximating tests for detecting if:
  - unit propagation can deduce all implied literals
  - equivalence reasoning can deduce all implied literals

$$\phi_{xor} \wedge \tilde{l}_1 \wedge \cdots \wedge \tilde{l}_k \quad \models_{up} \quad l$$
$$\phi_{xor} \wedge \tilde{l}_1 \wedge \cdots \wedge \tilde{l}_k \quad \models_{eq} \quad l$$

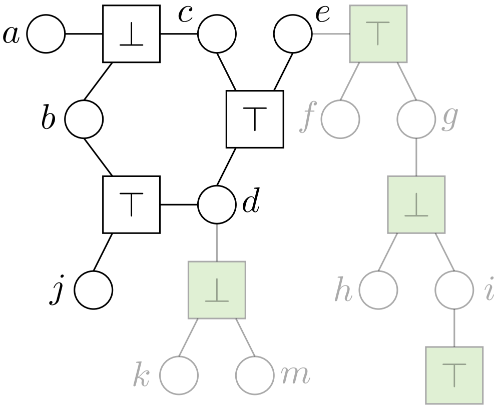
Tree-like Parity Constraints



$$
\begin{aligned}
a \oplus b \oplus c &\equiv \bot \\
b \oplus d \oplus j &\equiv \top \\
c \oplus d \oplus e &\equiv \top \\
d \oplus k \oplus m &\equiv \bot \\
e \oplus f \oplus g &\equiv \top \\
g \oplus h \oplus i &\equiv \bot \\
i &\equiv \top
\end{aligned}
$$

- unit propagation can deduce all implied literals for "tree-like" parity constraints

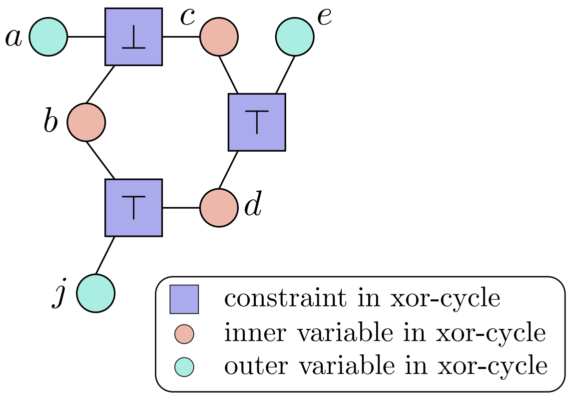
Tree-like Parity Constraints



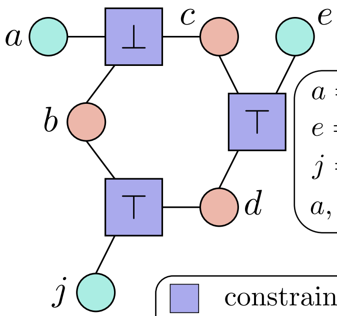
$$
\begin{aligned}
a \oplus b \oplus c &\equiv \bot \\
b \oplus d \oplus j &\equiv \top \\
c \oplus d \oplus e &\equiv \top \\
d \oplus k \oplus m &\equiv \bot \\
e \oplus f \oplus g &\equiv \top \\
g \oplus h \oplus i &\equiv \bot \\
i &\equiv \top
\end{aligned}
$$

- translate tree-like parity constraints to CNF and remove from xor-reasoning module

Parity Constraint (xor-)Cycles



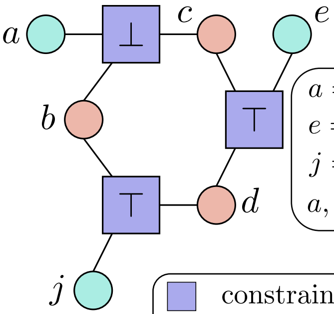
Parity Constraint (xor-)Cycles



$a = (b \neq c)$
$e = (c = d)$
$j = (b = d)$
a, e true $\rightsquigarrow j$ false

constraint in xor-cycle
inner variable in xor-cycle
outer variable in xor-cycle

Parity Constraint (xor-)Cycles

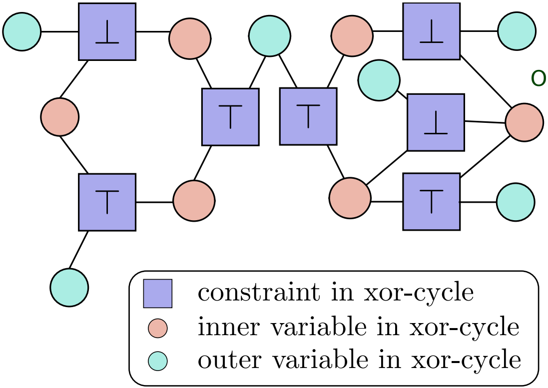


$$a = (b \neq c)$$
$$e = (c = d)$$
$$j = (b = d)$$
$$a, e \text{ true} \rightsquigarrow j \text{ false}$$

$$\begin{array}{l}
a \oplus b \oplus c \equiv \bot \\
b \oplus d \oplus j \equiv \top \\
c \oplus d \oplus e \equiv \top \\
\hline
a \oplus e \oplus j \equiv \bot
\end{array}$$

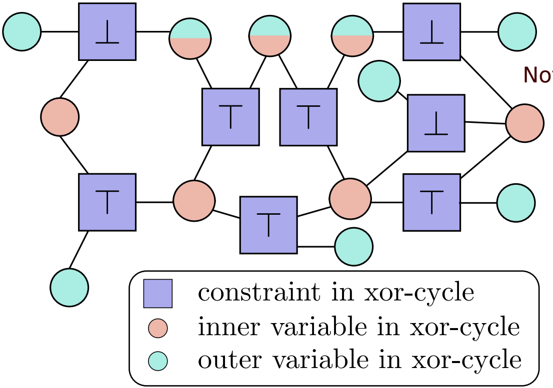
- constraint in xor-cycle (square)
- inner variable in xor-cycle (red circle)
- outer variable in xor-cycle (green circle)

Parity Constraint (xor-)Cycles



OK! cycle-partitionable

Parity Constraint (xor-)Cycles



Not cycle-partitionable!

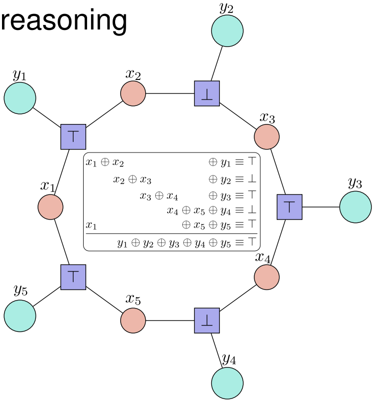
Classifying SAT Competition Instances

	SAT Competition 2005	2007	2009	2011
instances	857	376	573	1200
with xors⋆	123	100	140	111
unit propagation probably enough	19	10	18	15
tree-like	19	9	18	15
equivalence reasoning probably enough	20	21	52	40
cycle-partitionable	20	13	24	40

⋆ algorithm for xor pattern matching from CNF by M. Soos. SAT 2010

Simulating Equivalence Reasoning

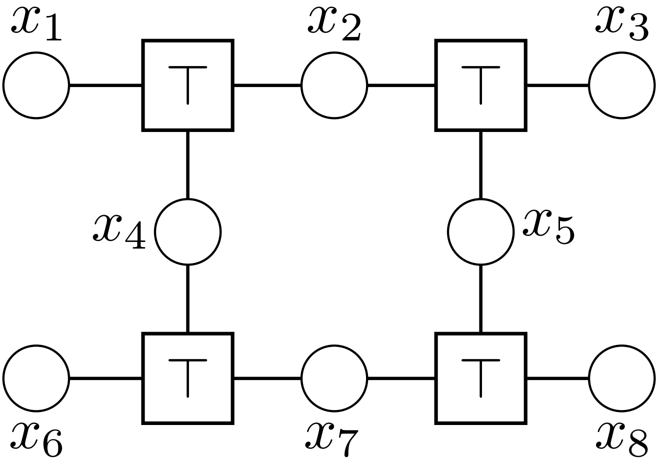
- connection between xor-cycles and equivalence reasoning can be exploited
- adding redundant parity constraint (linear combination) for each xor-cycle enables unit propagation to simulate equivalence reasoning



Simulating Equivalence Reasoning

- but there can be exponential number of xor-cycles!
- with extra variables, $O(n^3)$ additional parity constraints suffice
- in practice, much smaller number is needed
  - (see paper for details)

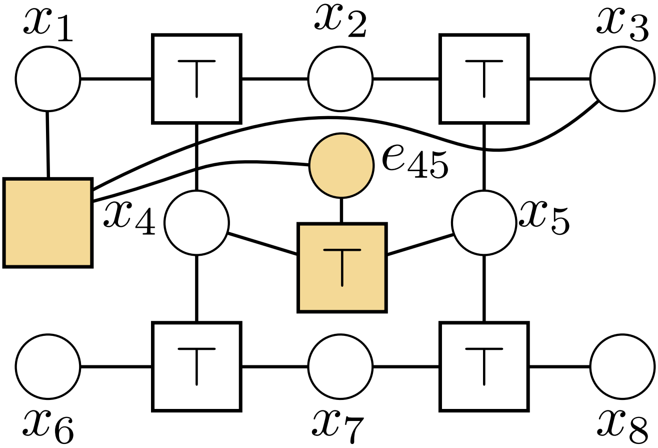
Simulating Equivalence Reasoning



$$\phi_{xor} \wedge x_1 \wedge x_3 \wedge x_6 \quad \models_{eq} \quad x_8$$

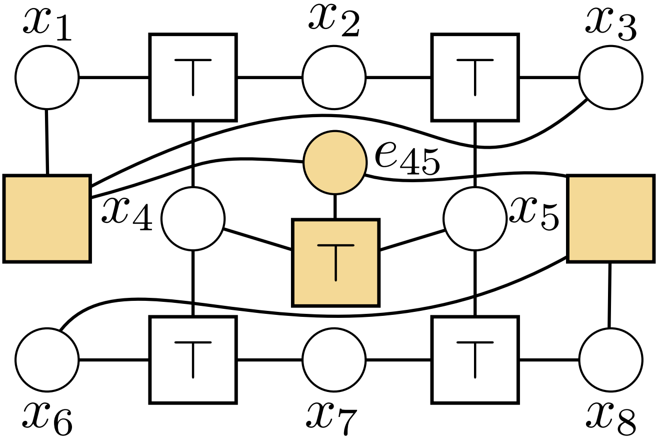
$$\phi_{xor} \wedge x_1 \wedge x_3 \wedge x_6 \quad \not\models_{up} \quad x_8$$

Simulating Equivalence Reasoning



$$\phi_{xor} \wedge x_1 \wedge x_3 \quad \models_{up} \quad e_{45}$$
$$\phi_{xor} \wedge e_{45} \wedge x_6 \quad \not\models_{up} \quad x_8$$

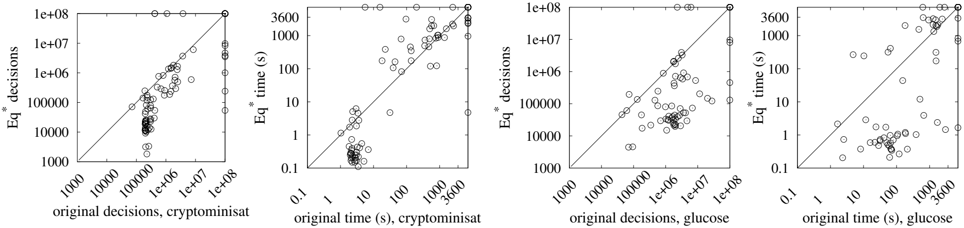
Simulating Equivalence Reasoning



$$\phi_{xor} \wedge x_1 \wedge x_3 \models_{up} e_{45}$$

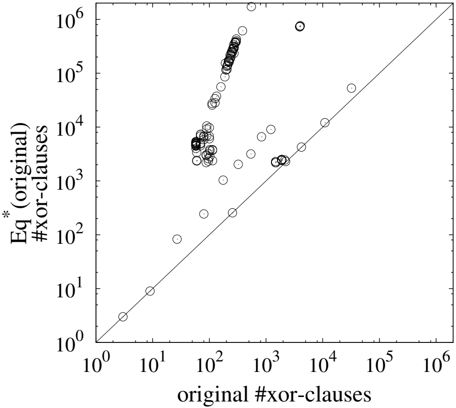
$$\phi_{xor} \wedge e_{45} \wedge x_6 \models_{up} x_8$$

Experimental Evaluation



- 123 SAT 2005 Competition instances with parity constraints
- x-axis = original instances
- y-axis = instances with additional parity constraints simulating equivalence reasoning with unit propagation
- cryptominisat 2.9.2 on the left, glucose 2.0 (SAT 2011 app. track winner) on the right
- significant reduction in decisions and often in solving time

Experimental Evaluation



- often manageable increase in instance size

Summary

- goal : effective SAT solver for cnf-xor SAT problem
- solution : xor-reasoning module integrateable to SAT solver
  - fast approximating tests for detecting whether unit propagation or equivalence reasoning is “enough”
  - tree-like parity constraints can be translated to CNF
  - strong connection between xor-cycles and equivalence reasoning
  - without extra variables, simulating equivalence reasoning requires exponential number of redundant constraints
  - with extra variables, unit propagation can simulate equivalence reasoning efficiently

Thank you for your attention

Questions?