Integration of Fourier-Motzkin based Variable-Elimination into iSAT

Karsten Scheibler, Felix Neubauer, Bernd Becker
Chair of Computer Architecture
SWIM 2012
Outline

What iSAT is

How iSAT works

Variable-Elimination
  Boolean Variable-Elimination
  Variable-Elimination based on Fourier-Motzkin
  One Constraint per Clause Form

Experimental Results

Conclusion & Future Work
What iSAT is

iSAT can be used as a (bounded) model checking tool for hybrid systems
What iSAT is

iSAT can be used as a (bounded) model checking tool for hybrid systems

Hybrid system: continuous and discrete dynamic behaviour
iSAT can be used as a (bounded) model checking tool for hybrid systems

Hybrid system: continuous and discrete dynamic behaviour

Model Checking: check if a model of a system satisfies desired properties
Examples
Examples

- **temp=22**
- **temp>24**
- **temp<20**
Examples

INIT

temp=22

ON

temp>24

OFF

temp<20

TRANS

temp=22 temp>24 OFF temp<20

SWIM 2012 Karsten Scheibler – Integration of Fourier-Motzkin based Variable-Elimination into iSAT 6 / 25
Examples

- **INIT**
  - temp = 22

- **ON**
  - temp > 24

- **OFF**
  - temp < 20

**TRANS**

- temp = 20
- temp = 22
- temp = 24

- **temp**
  - T₀₁, T₁₂, T₂₃, T₃₄, T₄₅

- **time**
Given by the model: INIT, TRANS
Bounded Model Checking

- Given by the model: INIT, TRANS
- Desired property: PROP

iSAT solves the following formulas:

\[ \text{INIT} \land \neg \text{PROP} \]

\[ \text{INIT} \land \text{TRANS}_{0,1} \land \neg \text{PROP} \]

\[ \text{INIT} \land \text{TRANS}_{0,1} \land \text{TRANS}_{1,2} \land \neg \text{PROP} \]

\[ \text{INIT} \land \text{TRANS}_{0,1} \land \text{TRANS}_{1,2} \land \text{TRANS}_{2,3} \land \neg \text{PROP} \]

\[ \vdots \]

If one formula is satisfiable the desired property was violated.
Bounded Model Checking

- Given by the model: INIT, TRANS
- Desired property: PROP
- iSAT solves the following formulas:
  - INIT ∧ ¬PROP
  - INIT ∧ TRANS_{0,1} ∧ ¬PROP
  - INIT ∧ TRANS_{0,1} ∧ TRANS_{1,2} ∧ ¬PROP
  - INIT ∧ TRANS_{0,1} ∧ TRANS_{1,2} ∧ TRANS_{2,3} ∧ ¬PROP
  - ...

- If one formula is satisfiable the desired property was violated
How iSAT works

- iSAT uses internally a conjunction of clauses
iSAT uses internally a conjunction of clauses

every clause is a disjunction of atoms
How iSAT works

- iSAT uses internally a conjunction of clauses
- every clause is a disjunction of atoms
- every atom is either:
  - a boolean variable: \( a \)
  - a negated boolean variable: \( \neg a \)
  - a simple bound: \( x < 5 \)
  - an arithmetic constraint: \( u = v^3, \ y = \sin(x), \ z = x + y, \ldots \)
How iSAT works

- iSAT uses internally a conjunction of clauses

- every clause is a disjunction of atoms

- every atom is either:
  - a boolean variable: \( a \)
  - a negated boolean variable: \( \neg a \)
  - a simple bound: \( x < 5 \)
  - an arithmetic constraint: \( u = v^3, y = \sin(x), z = x + y, \ldots \)

- every integer and real variable is bounded
Small example

\[ a, b \in \{\text{true, false}\}, \quad x \in [3, 7], y \in [-2, 49] \]

\[ (a \lor \neg b) \land (a \lor (y \leq 25)) \land (b \lor (y = x^2)) \]

\[ \text{Decision: } a = \text{false} \]

\[ \text{Deduction: } b = \text{false}, \quad y \leq 25, \quad y = x^2 \]

learn from conflicts already found (not shown in this example)
Small example

- \(a, b \in \{\text{true, false}\}, \quad x \in [3, 7], y \in [-2, 49]\)
  \[(a \lor \neg b) \land (a \lor (y \leq 25)) \land (b \lor (y = x^2))\]
- Decision: \(a = \text{false}\)
Small example

- \( a, b \in \{\text{true}, \text{false}\} \), \( x \in [3, 7], y \in [-2, 49] \)

\((a \lor \neg b) \land (a \lor (y \leq 25)) \land (b \lor (y = x^2))\)

- Decision: \( a = \text{false} \)

- Deduction: \( b = \text{false}, y \leq 25, y = x^2 \)
Small example

- $a, b \in \{\text{true}, \text{false}\}$, $x \in [3, 7], y \in [-2, 49]$

$$
(a \lor \neg b) \land (a \lor (y \leq 25)) \land (b \lor (y = x^2))
$$

- Decision: $a = \text{false}$

- Deduction: $b = \text{false}, y \leq 25, y = x^2$

- learn from conflicts already found (not shown in this example)
Small example (ICP)

\[ y = x^2: \]

\[ x \in [3, 7] \land y \in [-2, 49] \]

\[ \implies y \geq 9 \]

\[ x \in [3, 7] \land y \in [9, 25] \]

\[ \implies x \leq 5 \]
generalization of conflict-driven clause-learning (CDCL) framework as used in propositional SAT-Solvers
How iSAT works (2)

- Generalization of conflict-driven clause-learning (CDCL) framework as used in propositional SAT-Solvers
- Tight integration of interval constraint propagation (ICP) for arithmetic constraints into the solver core
How iSAT works (2)

- generalization of conflict-driven clause-learning (CDCL) framework as used in propositional SAT-Solvers
- tight integration of interval constraint propagation (ICP) for arithmetic constraints into the solver core
- use optimizations from propositional SAT-Solvers (non-chronological backtracking, two-watched literal scheme, restarts, ...)

SWIM 2012  Karsten Scheibler – Integration of Fourier-Motzkin based Variable-Elimination into iSAT
How iSAT works (3)

possible results:

- an unresolvable conflict is found (UNSATISFIABLE)
- all variables are point intervals (SATISFIABLE)
- intervals are small enough (CANDIDATE SOLUTION)
How iSAT works (3)

■ possible results:
  ■ an unresolvable conflict is found (UNSATISFIABLE)
  ■ all variables are point intervals (SATISFIABLE)
  ■ intervals are small enough (CANDIDATE SOLUTION)

■ iSAT will return a CANDIDATE SOLUTION for problems like $((x < y) \land (y < z) \land (z < x))$, because of interval arithmetic
possible results:
- an unresolvable conflict is found (UNSATISFIABLE)
- all variables are point intervals (SATISFIABLE)
- intervals are small enough (CANDIDATE SOLUTION)

iSAT will return a CANDIDATE SOLUTION for problems like \(((x < y) \land (y < z) \land (z < x))\), because of interval arithmetic

improve performance: solve more benchmarks and/or get more conclusive answers
Variable-Elimination

- used to reduce number of variables and clauses
- because smaller problems are solved faster usually
- today boolean variable elimination is a standard preprocessing technique in propositional SAT-Solvers
Boolean Variable-Elimination

- boolean formula in conjunctive normal form (CNF):
  \[ F_1 = (a \lor b \lor \neg c) \land (\neg a \lor b) \land (\neg a \lor c) \land (b \lor e) \]
Boolean Variable-Elimination

- boolean formula in conjunctive normal form (CNF):
  \[ F_1 = (a \lor b \lor \neg c) \land (\neg a \lor b) \land (\neg a \lor c) \land (b \lor e) \]

- want to eliminate boolean variable \( a \)

\[ F_2 = (b \lor \neg c) \land (b \lor e) \]

\( F_2 \) not equivalent to \( F_1 \), but:
\( F_2 \) satisfiable \( \iff \) \( F_1 \) satisfiable
Boolean Variable-Elimination

- boolean formula in conjunctive normal form (CNF):
  \[ F_1 = (a \lor b \lor \neg c) \land (\neg a \lor b) \land (\neg a \lor c) \land (b \lor e) \]
- want to eliminate boolean variable \( a \)
- do a full resolution (every \( a \)-clause with every \( \neg a \)-clause):
  - resolve\((a \lor b \lor \neg c), (\neg a \lor b)\) = \((b \lor \neg c \lor b) = (b \lor \neg c)\)
  - resolve\((a \lor b \lor \neg c), (\neg a \lor c)\) = \((b \lor \neg c \lor c) = \text{true}\)

\[ F_2 = (b \lor \neg c) \land (b \lor e) \]

\( F_2 \) not equivalent to \( F_1 \), but:
\[ F_2 \text{ satisfiable} \iff F_1 \text{ satisfiable} \]
Boolean Variable-Elimination

- boolean formula in conjunctive normal form (CNF):
  \[ F_1 = (a \lor b \lor \neg c) \land (\neg a \lor b) \land (\neg a \lor c) \land (b \lor e) \]

- want to eliminate boolean variable \( a \)

- do a full resolution (every \( a \)-clause with every \( \neg a \)-clause):
  \[
  \text{resolve}((a \lor b \lor \neg c), (\neg a \lor b)) = (b \lor \neg c \lor b) = (b \lor \neg c)
  \]
  \[
  \text{resolve}((a \lor b \lor \neg c), (\neg a \lor c)) = (b \lor \neg c \lor c) = \text{true}
  \]

- ignore clauses containing a tautology, in this example:
  \( (b \lor \neg c \lor c) \)
Boolean Variable-Elimination

- boolean formula in conjunctive normal form (CNF):
  \[ F_1 = (a \lor b \lor \neg c) \land (\neg a \lor b) \land (\neg a \lor c) \land (b \lor e) \]

- want to eliminate boolean variable \( a \)

- do a full resolution (every \( a \)-clause with every \( \neg a \)-clause):
  \[
  \text{resolve}((a \lor b \lor \neg c), (\neg a \lor b)) = (b \lor \neg c \lor b) = (b \lor \neg c) \\
  \text{resolve}((a \lor b \lor \neg c), (\neg a \lor c)) = (b \lor \neg c \lor c) = \text{true}
  \]

- ignore clauses containing a tautology, in this example:
  \( (b \lor \neg c \lor c) \)

- all other clauses are kept unchanged
Boolean Variable-Elimination

- boolean formula in conjunctive normal form (CNF):
  \[ F_1 = (a \lor b \lor \neg c) \land (\neg a \lor b) \land (\neg a \lor c) \land (b \lor e) \]

- want to eliminate boolean variable \( a \)

- do a full resolution (every \( a \)-clause with every \( \neg a \)-clause):
  
  \[
  \text{resolve}((a \lor b \lor \neg c), (\neg a \lor b)) = (b \lor \neg c \lor b) = (b \lor \neg c)
  \]
  
  \[
  \text{resolve}((a \lor b \lor \neg c), (\neg a \lor c)) = (b \lor \neg c \lor c) = \text{true}
  \]

- ignore clauses containing a tautology, in this example:
  \( b \lor \neg c \lor c \)

- all other clauses are kept unchanged

\[ F_2 = (b \lor \neg c) \land (b \lor e) \]
Boolean Variable-Elimination

- boolean formula in conjunctive normal form (CNF):
  \[ F_1 = (a \lor b \lor \neg c) \land (\neg a \lor b) \land (\neg a \lor c) \land (b \lor e) \]

- want to eliminate boolean variable \( a \)

- do a full resolution (every \( a \)-clause with every \( \neg a \)-clause):
  \[
  \text{resolve}((a \lor b \lor \neg c), (\neg a \lor b)) = (b \lor \neg c \lor b) = (b \lor \neg c) \\
  \text{resolve}((a \lor b \lor \neg c), (\neg a \lor c)) = (b \lor \neg c \lor c) = \text{true}
  \]

- ignore clauses containing a tautology, in this example:
  \((b \lor \neg c \lor c)\)

- all other clauses are kept unchanged

\[ F_2 = (b \lor \neg c) \land (b \lor e) \]

- \( F_2 \) not equivalent to \( F_1 \), but: \( F_2 \) satisfiable \( \iff \) \( F_1 \) satisfiable
Variable-Elimination based on Fourier-Motzkin

- rewrite equations and inequalities if needed to get $<$ or $\leq$ as relational operator
- conjunction of inequalities:
  \[ F_1 = (y < x) \land (x < v + w) \land (x < z) \land (v < u) \]
rewrite equations and inequalities if needed to get $<$ or $\leq$ as relational operator

conjunction of inequalities:

$F_1 = (y < x) \land (x < v + w) \land (x < z) \land (v < u)$

want to eliminate real variable $x$
Variable-Elimination based on Fourier-Motzkin

- rewrite equations and inequalities if needed to get $<$ or $\leq$ as relational operator
- conjunction of inequalities:
  $F_1 = (y < x) \land (x < v + w) \land (x < z) \land (v < u)$
- want to eliminate real variable $x$
- do a full resolution (every lower bound of $x$ with every upper bound of $x$):
  $\text{resolve}((y < x), (x < v + w)) = (y < v + w)$
  $\text{resolve}((y < x), (x < z)) = (y < z)$
Variable-Elimination based on Fourier-Motzkin

- rewrite equations and inequalities if needed to get $<$ or $\leq$ as relational operator
- conjunction of inequalities:
  $$F_1 = (y < x) \land (x < v + w) \land (x < z) \land (v < u)$$
- want to eliminate real variable $x$
- do a full resolution (every lower bound of $x$ with every upper bound of $x$):
  $$\text{resolve}((y < x), (x < v + w)) = (y < v + w)$$
  $$\text{resolve}((y < x), (x < z)) = (y < z)$$
- all other inequalities are kept unchanged
Variable-Elimination based on Fourier-Motzkin

- rewrite equations and inequalities if needed to get $<$ or $\leq$ as relational operator
- conjunction of inequalities:
  \[ F_1 = (y < x) \land (x < v + w) \land (x < z) \land (v < u) \]
- want to eliminate real variable $x$
- do a full resolution (every lower bound of $x$ with every upper bound of $x$):
  \[
  \text{resolve}((y < x), (x < v + w)) = (y < v + w)
  \]
  \[
  \text{resolve}((y < x), (x < z)) = (y < z)
  \]
- all other inequalities are kept unchanged
  \[
  F_2 = (y < v + w) \land (y < z) \land (v < u)
  \]
Variable-Elimination based on Fourier-Motzkin

- rewrite equations and inequalities if needed to get $<$ or $\leq$ as relational operator
- conjunction of inequalities:
  \[ F_1 = (y < x) \land (x < v + w) \land (x < z) \land (v < u) \]
- want to eliminate real variable $x$
- do a full resolution (every lower bound of $x$ with every upper bound of $x$):
  \[
  \text{resolve}((y < x), (x < v + w)) = (y < v + w)
  \]
  \[
  \text{resolve}((y < x), (x < z)) = (y < z)
  \]
- all other inequalities are kept unchanged
- \[ F_2 = (y < v + w) \land (y < z) \land (v < u) \]
- $F_2$ not equivalent to $F_1$, but: $F_2$ satisfiable $\iff F_1$ satisfiable
Variable-Elimination based on Fourier-Motzkin

Handling of $<$ and $\leq$:

- $y < x < z \implies y < z$
- $y < x \leq z \implies y < z$
- $y \leq x < z \implies y < z$
- $y \leq x \leq z \implies y \leq z$
But iSAT processes arbitrary boolean combinations of arithmetic constraints:

\[ F = (a \lor (y < x) \lor \neg b) \land ((x < v + w) \lor \neg c \lor (x < s \cdot z)) \land (v < \sin(u^2)) \land (\neg d \lor (x < 8) \lor e) \]

How to handle that?
But iSAT processes arbitrary boolean combinations of arithmetic constraints:

\[ F = (a \lor (y < x) \lor \neg b) \land ((x < v + w) \lor \neg c \lor (x < s \cdot z)) \land (v < \sin(u^2)) \land (\neg d \lor (x < 8) \lor e) \]

How to handle that?

\[\Rightarrow\] One Constraint per Clause Form (OCCF)
every clause may contain up to one arithmetic constraint
every clause may contain up to one arithmetic constraint
use boolean helper variables if needed:
- every clause may contain up to one arithmetic constraint
- use boolean helper variables if needed:

\[
F_1 = (a \lor (y < x) \lor \neg b) \land ((x < v + w) \lor \neg c \lor (x < s \cdot z)) \land \\
(v < \sin(u^2)) \land (\neg d \lor (x < 8) \lor e)
\]

\[
F_2 = (a \lor (y < x) \lor \neg b) \land ((x < v + w) \lor \neg c \lor h) \land \\
(\neg h \lor (x < s \cdot z)) \land (v < \sin(u^2)) \land (\neg d \lor (x < 8) \lor e)
\]
FM with OCCF

to eliminate an integer or real variable $x$ in an OCCF:

- do full resolution
FM with OCCF

to eliminate an integer or real variable $x$ in an OCCF:

- do full resolution
- resolve every clause containing a lower bound of $x$ with every clause containing an upper bound of $x$
to eliminate an integer or real variable $x$ in an OCCF:

- do full resolution
- resolve every clause containing a lower bound of $x$ with every clause containing an upper bound of $x$
- resolved clause contains
  - all boolean variables of both origin clauses
  - the result of resolve($x_{\text{lower}}$, $x_{\text{upper}}$)

\[(a \lor (y < x) \lor \neg b) \land ((x < v + w) \lor \neg c \lor h)\]
\[
\leadsto (a \lor \neg b \lor \neg c \lor h \lor (y < v + w))
\]
1. mark variables as “bad” if they occur in non-linear subexpressions like $sin(x), y \cdot z, \ldots$
Implementation in iSAT

1. mark variables as “bad” if they occur in non-linear subexpressions like $\sin(x), y \cdot z, \ldots$
2. select a “good” variable and try to eliminate it
Implementation in iSAT

1. Mark variables as “bad” if they occur in non-linear subexpressions like $\sin(x), y \cdot z, \ldots$
2. Select a “good” variable and try to eliminate it
3. If this variable occurs in too many clauses, skip it
Implementation in iSAT

1. mark variables as “bad” if they occur in non-linear subexpressions like $\sin(x), y \cdot z, \ldots$
2. select a “good” variable and try to eliminate it
3. if this variable occurs in too many clauses, skip it
4. do full resolution (ignore clauses containing a tautology)
Implementation in iSAT

1. mark variables as “bad” if they occur in non-linear subexpressions like $\sin(x)$, $y \cdot z$, …
2. select a “good” variable and try to eliminate it
3. if this variable occurs in too many clauses, skip it
4. do full resolution (ignore clauses containing a tautology)
5. count resulting clauses, discard if too many clauses
Implementation in iSAT

1. mark variables as “bad” if they occur in non-linear subexpressions like $\sin(x), y \cdot z, \ldots$
2. select a “good” variable and try to eliminate it
3. if this variable occurs in too many clauses, skip it
4. do full resolution (ignore clauses containing a tautology)
5. count resulting clauses, discard if too many clauses
6. otherwise replace original clauses with the resolved clauses
mark variables as “bad” if they occur in non-linear subexpressions like $\sin(x), y \cdot z, \ldots$

select a “good” variable and try to eliminate it

if this variable occurs in too many clauses, skip it

do full resolution (ignore clauses containing a tautology)

count resulting clauses, discard if too many clauses

otherwise replace original clauses with the resolved clauses

goto 2
Experimental Results

We wanted:

- to solve more benchmarks
- to have more conclusive answers (SATISFIABLE or UNSATISFIABLE)
Experimental Results

We got:

more solved benchmarks and more conclusive answers!

<table>
<thead>
<tr>
<th></th>
<th>without FM-VE</th>
<th>with FM-VE</th>
<th>+/-</th>
</tr>
</thead>
<tbody>
<tr>
<td>SATISFIABLE</td>
<td>6</td>
<td>8</td>
<td>+2</td>
</tr>
<tr>
<td>UNSATISFIABLE</td>
<td>136</td>
<td>146</td>
<td>+10</td>
</tr>
<tr>
<td>CANDIDATE SOLUTION</td>
<td>256</td>
<td>253</td>
<td>-3</td>
</tr>
<tr>
<td>Σ</td>
<td>398</td>
<td>407</td>
<td>+9</td>
</tr>
</tbody>
</table>

(overall 543 Benchmarks, Timeout 900 seconds)
Conclusion & Future Work

- solved more benchmarks
- got more conclusive answers
- currently variables are eliminated in order of appearance, examine heuristics for a better selection
- include subsumption checks as used in propositional SAT-Solvers
- compare Fourier-Motzkin VE to Loos-Weispfenning VE
References
