A Generalized Two-watched-literal Scheme in a mixed Boolean and Non-linear Arithmetic Constraint Solver[1]

Tino Teige[*] Christian Herde[*] Martin Fränzle[*]
Natalia Kalinnik[†] Andreas Eggers[*]

[*] University of Oldenburg, Germany

[†] University of Freiburg, Germany

STCS @ EPIA, Guimarães, Portugal
December 7, 2007

[1] partly supported by the German Research Council (DFG) as part of the AVACS project (`www.avacs.org`)

Aim of the talk

$$(\neg b \lor y > \sin(x) \lor a \leq -2.76) \land (x = y \cdot z \lor x = 1/y)$$

Adapting the **two-watched literal scheme** from the propositional case to the **mixed Boolean and non-linear arithmetic** framework.

Recalling the two-watched literal scheme

Idea: Accelerating unit propagation by lazy evaluation of clauses

two watched non-false literals as witness for non-unitness

- $(\neg \mathbf{a} \vee \mathbf{b} \vee c \vee \neg d \vee \neg e \vee \neg f)$
- $\neg c$, $\neg b$
- $(\neg \mathbf{a} \vee b \vee c \vee \neg \mathbf{d} \vee \neg e \vee \neg f)$
- e, f, a
- $(\neg a \vee b \vee c \vee \neg \mathbf{d} \vee \neg e \vee \neg f)$
- deducing $\neg d$

Recalling the two-watched literal scheme

Idea: Accelerating unit propagation by lazy evaluation of clauses
two watched non-false literals as witness for non-unitness

- $(\neg \mathbf{a} \vee \mathbf{b} \vee c \vee \neg d \vee \neg e \vee \neg f)$
- $\neg c$, $\neg b$
- $(\neg \mathbf{a} \vee b \vee c \vee \neg \mathbf{d} \vee \neg e \vee \neg f)$
- e, f, a
- $(\neg a \vee b \vee c \vee \neg \mathbf{d} \vee \neg e \vee \neg f)$
- deducing $\neg d$

Recalling the two-watched literal scheme

Idea: Accelerating unit propagation by lazy evaluation of clauses
two watched non-false literals as witness for non-unitness

- $(\neg \mathbf{a} \vee \mathbf{b} \vee c \vee \neg d \vee \neg e \vee \neg f)$
- $\neg c$, $\neg b$
- $(\neg \mathbf{a} \vee b \vee c \vee \neg \mathbf{d} \vee \neg e \vee \neg f)$
- e, f, a
- $(\neg a \vee b \vee c \vee \neg \mathbf{d} \vee \neg e \vee \neg f)$
- deducing $\neg d$

Recalling the two-watched literal scheme

Idea: Accelerating unit propagation by lazy evaluation of clauses
two watched non-false literals as witness for non-unitness

- $(\neg \mathbf{a} \vee \mathbf{b} \vee c \vee \neg d \vee \neg e \vee \neg f)$
- $\neg c$, $\neg b$
- $(\neg \mathbf{a} \vee b \vee c \vee \neg \mathbf{d} \vee \neg e \vee \neg f)$
- e, f, a
- $(\neg a \vee b \vee c \vee \neg \mathbf{d} \vee \neg e \vee \neg f)$
- deducing $\neg d$

Recalling the two-watched literal scheme

Idea: Accelerating unit propagation by lazy evaluation of clauses two watched non-false literals as witness for non-unitness

- $(\neg \mathbf{a} \vee \mathbf{b} \vee c \vee \neg d \vee \neg e \vee \neg f)$
- $\neg c$, $\neg b$
- $(\neg \mathbf{a} \vee b \vee c \vee \neg \mathbf{d} \vee \neg e \vee \neg f)$
- e, f, a
- $(\neg a \vee b \vee c \vee \neg \mathbf{d} \vee \neg e \vee \neg f)$
- deducing $\neg d$

Recalling the two-watched literal scheme

Idea: Accelerating unit propagation by lazy evaluation of clauses two watched non-false literals as witness for non-unitness

- $(\neg \mathbf{a} \vee \mathbf{b} \vee c \vee \neg d \vee \neg e \vee \neg f)$
- $\neg c$, $\neg b$
- $(\neg \mathbf{a} \vee b \vee c \vee \neg \mathbf{d} \vee \neg e \vee \neg f)$
- e, f, a
- $(\neg a \vee b \vee c \vee \neg \mathbf{d} \vee \neg e \vee \neg f)$
- deducing $\neg d$

Recalling the two-watched literal scheme

Idea: Accelerating unit propagation by lazy evaluation of clauses
two watched non-false literals as witness for non-unitness

- $(\neg \mathbf{a} \vee \mathbf{b} \vee c \vee \neg d \vee \neg e \vee \neg f)$
- $\neg c$, $\neg b$
- $(\neg \mathbf{a} \vee b \vee c \vee \neg \mathbf{d} \vee \neg e \vee \neg f)$
- e, f, a
- $(\neg a \vee b \vee c \vee \neg \mathbf{d} \vee \neg e \vee \neg f)$
- deducing $\neg d$

Recalling the two-watched literal scheme

Idea: Accelerating unit propagation by lazy evaluation of clauses
two watched non-false literals as witness for non-unitness

- $(\neg \mathbf{a} \vee \mathbf{b} \vee c \vee \neg d \vee \neg e \vee \neg f)$
- $\neg c$, $\neg b$
- $(\neg \mathbf{a} \vee b \vee c \vee \neg \mathbf{d} \vee \neg e \vee \neg f)$
- e, f, a
- $(\neg a \vee b \vee c \vee \neg \mathbf{d} \vee \neg e \vee \neg f)$
- deducing $\neg d$

Recalling the two-watched literal scheme

Idea: Accelerating unit propagation by lazy evaluation of clauses two watched non-false literals as witness for non-unitness

- $(\neg \mathbf{a} \vee \mathbf{b} \vee c \vee \neg d \vee \neg e \vee \neg f)$
- $\neg c, \neg b$
- $(\neg \mathbf{a} \vee b \vee c \vee \neg \mathbf{d} \vee \neg e \vee \neg f)$
- e, f, a
- $(\neg a \vee b \vee c \vee \neg \mathbf{d} \vee \neg e \vee \neg f)$
- deducing $\neg d$

Recalling the two-watched literal scheme

Idea: Accelerating unit propagation by lazy evaluation of clauses
two watched non-false literals as witness for non-unitness

- $(\neg \mathbf{a} \vee \mathbf{b} \vee c \vee \neg d \vee \neg e \vee \neg f)$
- $\neg c$, $\neg b$
- $(\neg \mathbf{a} \vee {\color{red} b} \vee {\color{red} c} \vee \neg \mathbf{d} \vee \neg e \vee \neg f)$
- e, f, a
- $({\color{red} \neg a} \vee {\color{red} b} \vee {\color{red} c} \vee \neg \mathbf{d} \vee {\color{red} \neg e} \vee {\color{red} \neg f})$
- deducing $\neg d$

iSAT algorithm [Fränzle et al 06/07]

- tackles **conjunction** of **disjunctions** of **(arithmetic) atoms** over the **reals**, integers, Booleans, e.g.

  $$(\neg b \vee y > \sin(x) \vee a \leq -2.76) \;\wedge\; (x = y \cdot z \vee x = 1/y)$$

  Note: For each mixed Boolean arithmetic formula there is an equi-satisfiable linearly-sized formula in such CNF!
- **generalization of DPLL:**
  - manipulating *interval valuations*
  - *branching* (splitting intervals)
  - *propagation* (unit + interval propagation)
  - *conflict-driven clause learning*
  - *non-chronological backtracking*

iSAT algorithm [Fränzle et al 06/07]

- tackles **conjunction** of **disjunctions** of **(arithmetic) atoms** over the **reals**, integers, Booleans, e.g.

$$(\neg b \vee y > \sin(x) \vee a \leq -2.76) \ \wedge \ (x = y \cdot z \vee x = 1/y)$$

  Note: For each mixed Boolean arithmetic formula there is an equi-satisfiable linearly-sized formula in such CNF!

- **generalization of DPLL:**
  - manipulating *interval valuations*
  - *branching* (splitting intervals)
  - *propagation* (unit + interval propagation)
  - *conflict-driven clause learning*
  - *non-chronological backtracking*

HySAT tool

- iSAT as core algorithm (moreover, bounded model checker for reachability analysis of hybrid systems, `http://hysat.informatik.uni-oldenburg.de/`)
- reads **arbitrary** Boolean combination of **arbitrary** non-linear constraints, e.g. $(x > y \implies y^2 \geq \sin(x))$
- rewrites into CNF (by Tseitin-like transformation)

$$(x > y \implies y^2 \geq \sin(x)) \rightsquigarrow (x \leq y \vee h \geq \sin(x)) \wedge (h = y^2)$$

Two-watched atom scheme

- 2 atoms watched in each clause

$$(\underline{x \leq \sin(y)} \vee \underline{y > x^2} \vee y \leq -3.1 \vee \neg b)$$

- visit clause whenever **new interval for a variable** could change truth value of one watch, e.g.
  - $x \geq 0$ then visit clause
  - $b = \texttt{true}$ $(b \geq 1)$ do not visit clause
- if a watched atom evaluates to **false** (i.e. each possible combination of values do **not satisfy** atom) then search for a replacement

Two-watched atom scheme

- 2 atoms watched in each clause

$$(\underline{x \leq \sin(y)} \vee \underline{y > x^2} \vee y \leq -3.1 \vee \neg b)$$

- visit clause whenever **new interval for a variable** could change truth value of one watch, e.g.
  - $x \geq 0$ then visit clause
  - $b = \texttt{true}$ $(b \geq 1)$ do not visit clause
- if a watched atom evaluates to **false** (i.e. each possible combination of values do **not satisfy** atom) then search for a replacement

Two-watched atom scheme

- 2 atoms watched in each clause

$$(\underline{x \leq \sin(y)} \vee \underline{y > x^2} \vee y \leq -3.1 \vee \neg b)$$

- visit clause whenever **new interval for a variable** could change truth value of one watch, e.g.
  - $x \geq 0$ then visit clause
  - $b = \texttt{true}$ $(b \geq 1)$ do not visit clause
- if a watched atom evaluates to **false** (i.e. each possible combination of values do **not satisfy** atom) then search for a replacement

Two-watched atom scheme: Example

- $(\underline{x \le \sin(y)} \vee \underline{y > x^2} \vee y \le -3.1 \vee \neg b)$,

  $x \in (-3, 8],\ y \in [-4, 53),\ b \in \mathbb{B} = \{0, 1\}$
- $b \ge 1$: no visit
- $x > 6.2$: $x \le \sin(y)$ false, since $x \le \sin(y) \le 1$
- $(x \le \sin(y) \vee \underline{y > x^2} \vee \underline{y \le -3.1} \vee \neg b)$,

  $x \in (6.2, 8],\ y \in [-4, 53),\ b \in \{1\}$
- $y < 47$: visit clause, no watch violated
- $y \ge -2$: $y \le -3.1$ false
- $(x \le \sin(y) \vee y > x^2 \vee y \le -3.1 \vee \neg b)$
- clause becomes **unit** propagating $y > x^2$, which deduces, e.g., $\mathbf{y} > x^2 > 6.2^2 = \mathbf{38.44}$

Two-watched atom scheme: Example

- $(\underline{x \leq \sin(y)} \vee \underline{y > x^2} \vee y \leq -3.1 \vee \neg b)$,

  $x \in (-3, 8],\ y \in [-4, 53),\ b \in \mathbb{B} = \{0, 1\}$
- $b \geq 1$: no visit
- $x > 6.2$: $x \leq \sin(y)$ false, since $x \leq \sin(y) \leq 1$
- $(x \leq \sin(y) \vee \underline{y > x^2} \vee \underline{y \leq -3.1} \vee \neg b)$,

  $x \in (6.2, 8],\ y \in [-4, 53),\ b \in \{1\}$
- $y < 47$: visit clause, no watch violated
- $y \geq -2$: $y \leq -3.1$ false
- $(x \leq \sin(y) \vee y > x^2 \vee y \leq -3.1 \vee \neg b)$
- clause becomes **unit** propagating $y > x^2$, which deduces, e.g., $\mathbf{y} > x^2 > 6.2^2 = \mathbf{38.44}$

Two-watched atom scheme: Example

- $(\underline{x \le \sin(y)} \vee \underline{y > x^2} \vee y \le -3.1 \vee \neg b)$,
  $$x \in (-3, 8],\ \ y \in [-4, 53),\ \ b \in \mathbb{B} = \{0, 1\}$$
- $b \ge 1$: no visit
- $x > 6.2$: $x \le \sin(y)$ false, since $x \le \sin(y) \le 1$
- $(x \le \sin(y) \vee \underline{y > x^2} \vee \underline{y \le -3.1} \vee \neg b)$,
  $$x \in (6.2, 8],\ \ y \in [-4, 53),\ \ b \in \{1\}$$
- $y < 47$: visit clause, no watch violated
- $y \ge -2$: $y \le -3.1$ false
- $(x \le \sin(y) \vee y > x^2 \vee y \le -3.1 \vee \neg b)$
- clause becomes **unit** propagating $y > x^2$, which deduces, e.g., $\mathbf{y} > x^2 > 6.2^2 = \mathbf{38.44}$

Two-watched atom scheme: Example

- $(\underline{x \leq \sin(y)} \vee \underline{y > x^2} \vee y \leq -3.1 \vee \neg b)$,
  $$x \in (-3, 8],\ \ y \in [-4, 53),\ \ b \in \mathbb{B} = \{0, 1\}$$
- $b \geq 1$: no visit
- $x > 6.2$: $x \leq \sin(y)$ false, since $x \leq \sin(y) \leq 1$
- $(x \leq \sin(y) \vee \underline{y > x^2} \vee \underline{y \leq -3.1} \vee \neg b)$,
  $$x \in (6.2, 8],\ \ y \in [-4, 53),\ \ b \in \{1\}$$
- $y < 47$: visit clause, no watch violated
- $y \geq -2$: $y \leq -3.1$ false
- $(x \leq \sin(y) \vee y > x^2 \vee y \leq -3.1 \vee \neg b)$
- clause becomes **unit** propagating $y > x^2$, which deduces, e.g., $\mathbf{y} > x^2 > 6.2^2 = \mathbf{38.44}$

Two-watched atom scheme: Example

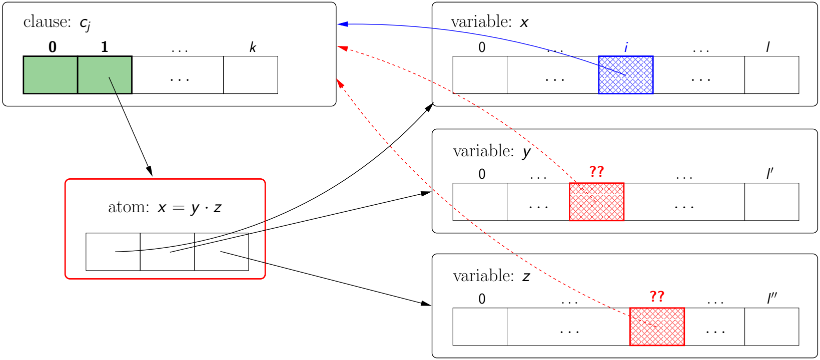
- $(\underline{x \leq \sin(y)} \vee \underline{y > x^2} \vee y \leq -3.1 \vee \neg b)$,
  $$x \in (-3, 8],\ \ y \in [-4, 53),\ \ b \in \mathbb{B} = \{0, 1\}$$
- $b \geq 1$: no visit
- $x > 6.2$: $x \leq \sin(y)$ false, since $x \leq \sin(y) \leq 1$
- $(x \leq \sin(y) \vee \underline{y > x^2} \vee \underline{y \leq -3.1} \vee \neg b)$,
  $$x \in (6.2, 8],\ \ y \in [-4, 53),\ \ b \in \{1\}$$
- $y < 47$: visit clause, no watch violated
- $y \geq -2$: $y \leq -3.1$ false
- $(x \leq \sin(y) \vee y > x^2 \vee y \leq -3.1 \vee \neg b)$
- clause becomes **unit** propagating $y > x^2$, which deduces, e.g., $\mathbf{y} > x^2 > 6.2^2 = \mathbf{38.44}$

Clause evaluation: Possible cases

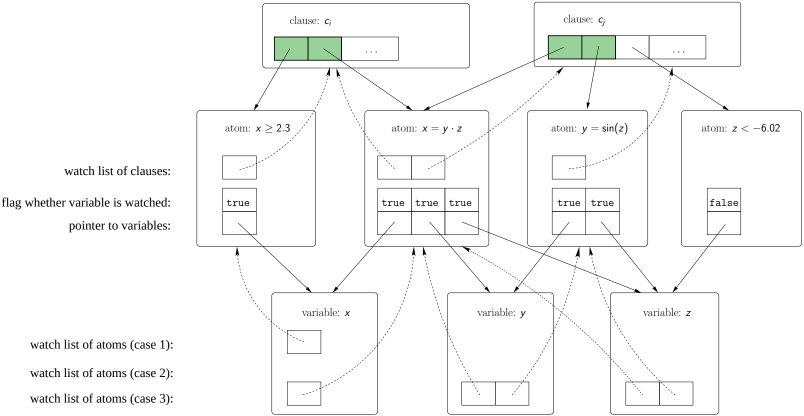
# false watches	1 new watch	2 new watches	action
0	—	—	nothing
1	yes	—	set new watch
	no	—	propagate other watch
2	—	yes	set new watches
	yes	no	propagate new watch
	no	—	conflict analysis

Updating watch lists

Problem: Avoid performing a time-consuming list search.

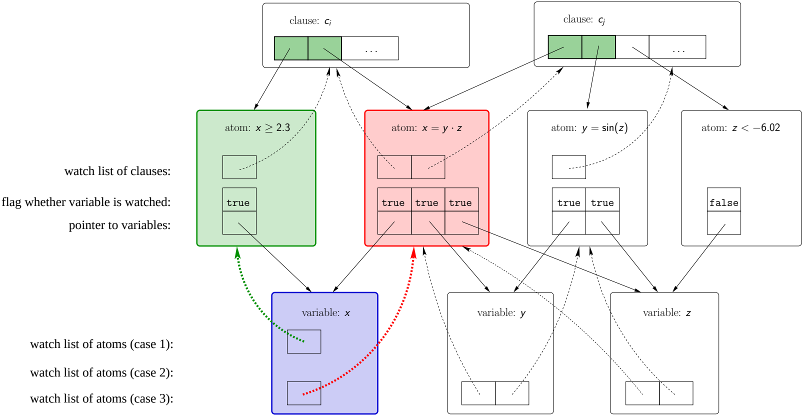


Data-structure & two-level watch scheme



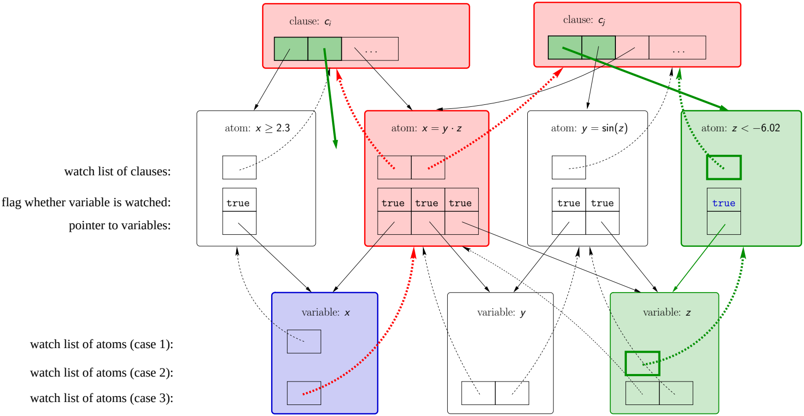
Two-level watch scheme I

New upper bound $x < 3.5$: $x = y \cdot z$ becomes false.



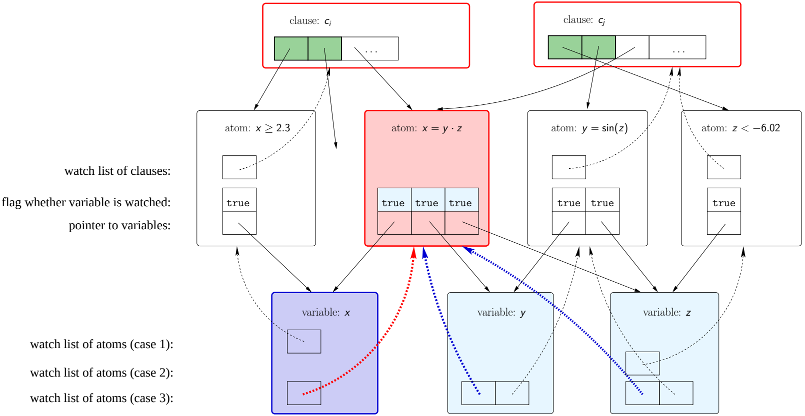
Two-level watch scheme II

Visit clauses in WL for $x = y \cdot z$, look for new watches.



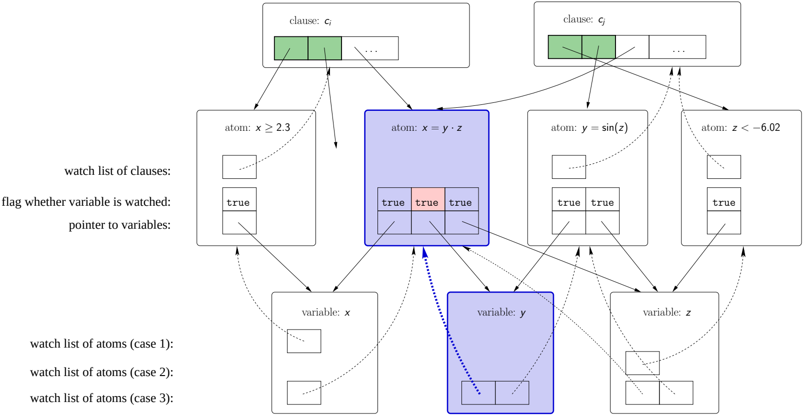
Two-level watch scheme III

WL for $x = y \cdot z$ updated, but **not** for its variables x, y, z.



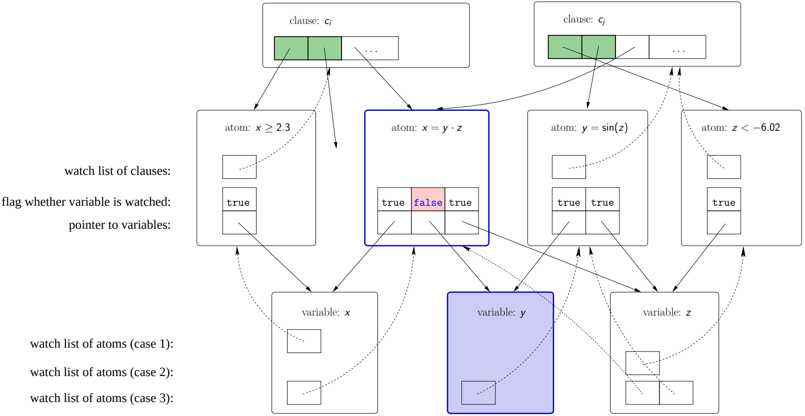
Two-level watch scheme IV

Update WLs for variables **lazily** when new bounds processed.



Two-level watch scheme V

$x = y \cdot z$ unwatched, set flag for y to `false`, update WLs for y.



Empirical results

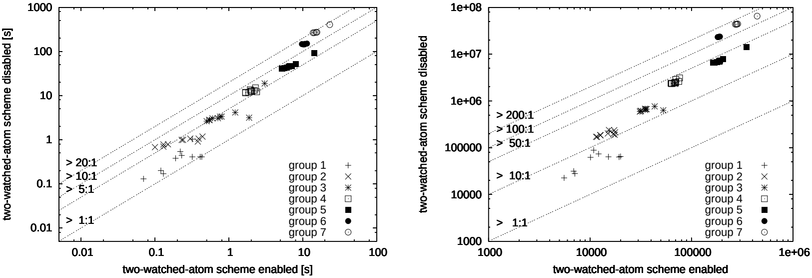


Figure: Performance impact of the two-watched-atom scheme: runtime in seconds (left) and number of clause evaluations (right)

Future and ongoing work

1. **Efficiency:** accelerating iSAT
   - (variable-selection and interval-splitting) *decision heuristics*
   - *numerical methods* (interval Newton, linear programming)
   - *parallelization* (divide-and-conquer, different BMC instances)
   - *low-level code optimizations* for improving cache behavior
   - ...
2. **Extension:** handling broader classes of constraints
   - constraints containing *ordinary differential equations*
   - *existential & stochastic quantification* of discrete variables
   - *generation of Craig interpolants*

Thank you!

iSAT algorithm

```
iSAT() {
    while (true) {
        while (true) {
            result = deduce();                  // Deducing.
            if (result == CONFLICT) {
                // Learning & Backjumping.
                resolved = analyze_conflict();
                if (!resolved) {
                    return "UNSATISFIABLE";
                }
            }
            else if (result == SOLUTION) {
                return "SATISFIABLE";
            }
            else {
                break;
            }
        }
        if (!decide_next_branch()) {        // Branching.
            return "UNKNOWN";
        }
    }
}
```

HySAT tool

- **old** version just supports *watching of simple bounds*, **not** of arithmetic atoms
- E.g., $(x > y \implies y^2 \geq \sin(x))$:

$$\begin{aligned} x > y \quad & \rightsquigarrow \quad (h_1 > 0) \wedge (h_1 = x - y) \\ y^2 \geq \sin(x) \quad & \rightsquigarrow \quad (h_4 \geq 0) \wedge (h_2 = y^2) \wedge \\ & \qquad (h_3 = \sin(x)) \wedge (h_4 = h_2 - h_3) \end{aligned}$$

  Rewritten formula: $(h_1 \leq 0 \vee h_4 \geq 0) \wedge (h_1 = x - y) \wedge \ldots$
- two types of clauses:
  1. **disjunctive** clauses contain **simple** bounds only
  2. **arithmetic** atoms occur only in **single** clauses

Motivation of watching arithmetic atoms I

$$(x > y \implies y^2 \geq \sin(x)) \quad \rightsquigarrow \quad (x \leq y \vee h \geq \sin(x)) \wedge (h = y^2)$$

- homogeneous representation of clauses
- saving auxiliary variables
- learned clauses not restricted to just simple bounds
- (potentially) saving expensive interval computations

Motivation of watching arithmetic atoms II

- direct handling of partial functions/operators

$$(x/y > 2.3 \vee \ldots) \quad \text{and} \quad (h > 2.3 \vee \ldots) \wedge (h = x/y)$$

are **not** satisfiability-equivalent since we **exclude** $y = 0$ for which potentially a solution exists

Motivation of watching arithmetic atoms III

- (generalized) polarity optimization by Tseitin-transformation

given:	$(x \cdot y \cdot z \geq -0.7)$	
old:	$(h' \geq -0.7)$	$\wedge \; (h' = x \cdot h)$
		$\wedge \; (h = y \cdot z)$
new:	$(x \cdot h \geq -0.7)$	$\wedge \; (x < 0 \vee h \leq y \cdot z)$
		$\wedge \; (x \geq 0 \vee h \geq y \cdot z)$

$x \cdot y \cdot z \geq x \cdot h \geq -0.7$,

$$x \cdot y \cdot z \geq x \cdot h \;\text{ iff }\; \begin{cases} y \cdot z \geq h & ; x \geq 0 \\ y \cdot z \leq h & \text{otherwise} \end{cases}$$

  - positive effect on deciding satisfiability
  - facilitating improved watching and less interval computations

Empirical results I

- *Comparison:* two-watching **enabled** vs. **disabled** (simulates old version in a certain sense) under same syntactic formulae
- *Why not against old version?* HySAT has a strong preprocessing. Syntactic representation (with sharing common subexpressions) of a formula potentially strongly influence solving (result, runtime)!

  $$(x>y) \wedge (x<y) \quad \text{vs.} \quad (h>0) \wedge (h<0) \wedge (h = x - y)$$

  Alternative comparisons with old HySAT
  - *either* on the same syntactic formulae but then **no watching of arithmetic atoms**
  - *or* on different syntactic formulae but then empirical results **strongly depend on concrete representation**.

  I.e., both alternatives are **not** informative of the impact of watching arithmetic atoms.

Empirical results II

- *Why random benchmarks?* **Scalable** in **number of variables** and **clauses**, and **size of clauses**. Concerning 2-watching, structure of a formula is not as important as for heuristics like decision strategies or learning.