About the SAT Solvers

Limmat, Compsat, Funex

and the QBF Solver

Quantor

May 2003

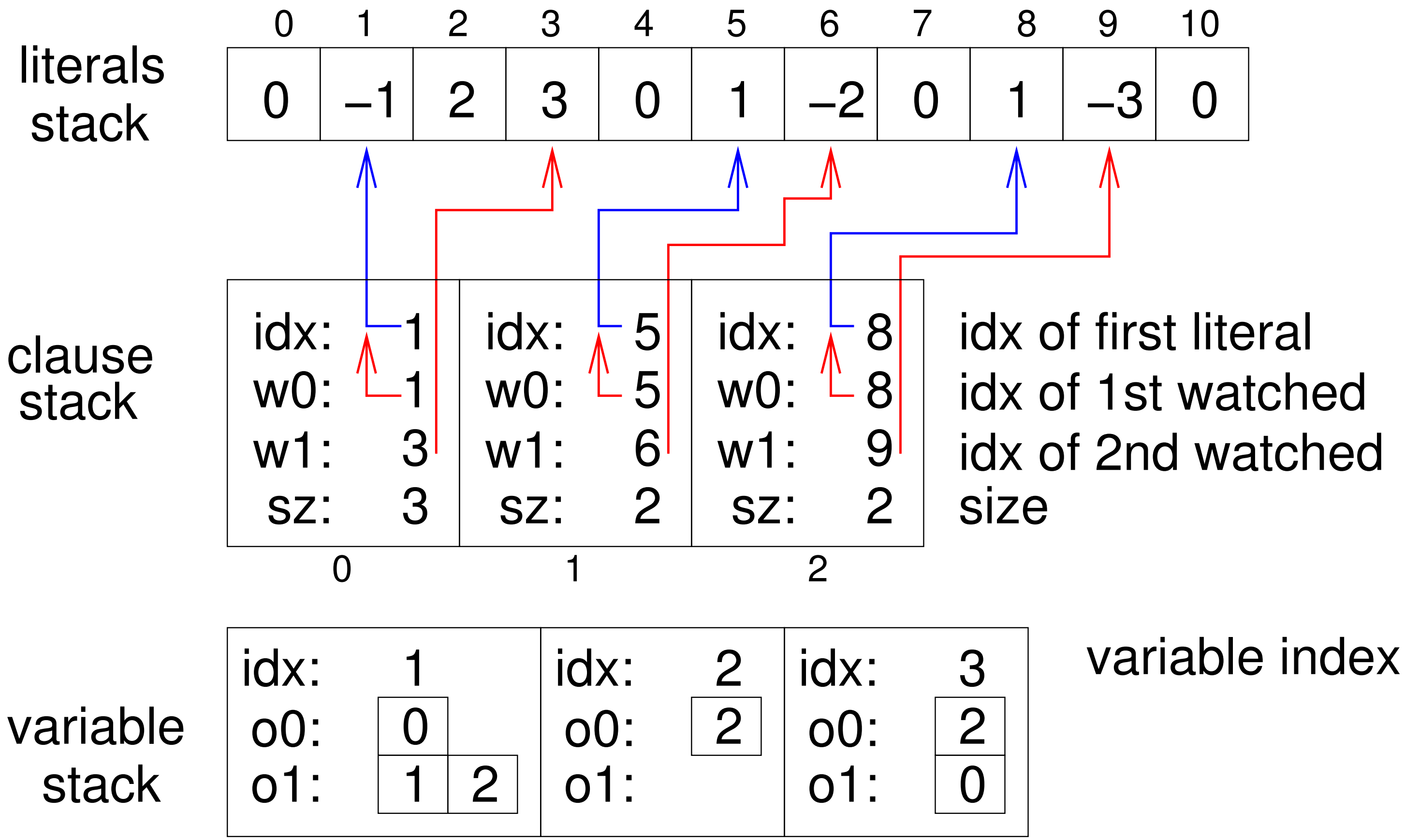
Armin Biere

Computer Systems Institute

ETH Zürich, Switzerland

SAT'03, Santa Margherita Ligure, Portofino, Italy

Separate Clause Structure



o0/o1 are stacks of clause indices in which resp. literal is watched

Separate Clause Structure

- implemented in both Limmat and Funex but not in Compsat

- 1st **benefit**: direct access to other watched literal

  - no traversal when assigning literals where the other watched is satisfied

- 2nd **benefit**: clause information accessible in BCP

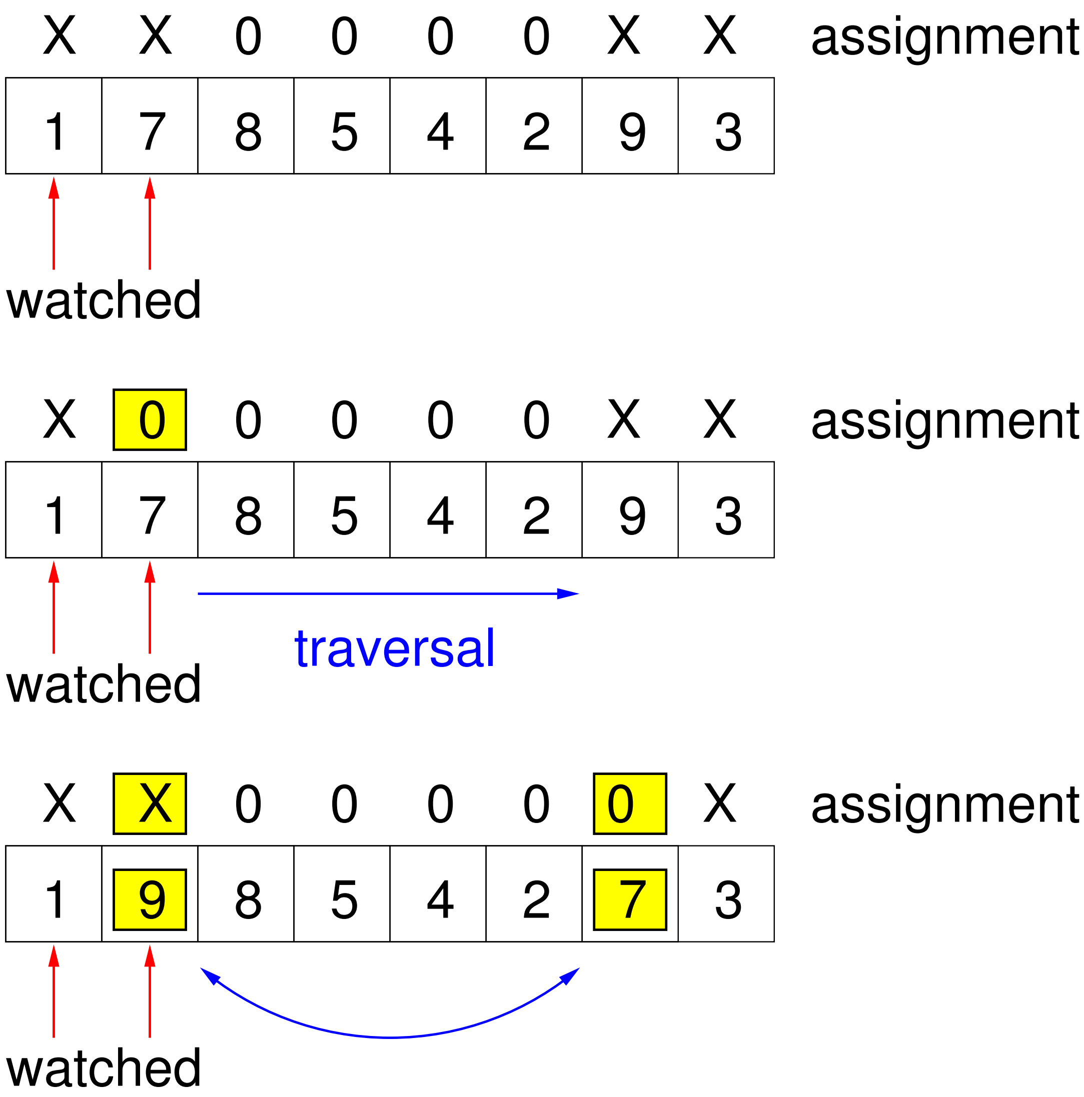
  - heuristics: shorter clauses as reasons in assignments preferred

- major **drawback**: additional indirection in occurrence lookup

  - Zchaff and Compsat: occurrence stacks store literal positions directly

Compsat: Watched-Literals Approach

eg [VanGelder]



Compsat: Additional Features

- van Gelder's approach allows very simple data structures (integers and integer stacks)

- **comp**act memory layout
  - as in BDD library ABCD: **minimize time by minimizing space**

- various space optimized integer stacks
  - compact stack with 8 Byte anchor (C++ STL requires 12 Byte anchor)
  - elements can be 16 bit or 32 bit (may change dynamically)
  - fully configurable

$\Rightarrow$ **therefore very low memory footprint**

Compsat: Flaw

- submitted version of Compsat had a serious last minute bug
- **BCP queue was not flushed after restart**
- showed up in large benchmarks only
  - at least one restart required
- bug escaped automated test suite
- one line bug fix, since flushing of BCP queue already implemented
  - added new test cases with restart intervals of length 1

Funex Additional Features

- incorporated **Berkmin** style decision function (clause linking)
  - cache of satisfied clauses reduces time spent in decision function
- selection of decision functions is specified as ω-regular expression
  - decision functions: `dlis`, `horn`, `chaff`, `berkmin`
  - default selection: `(horn.berkmin^3000)^infinity`
- dedicated BCP for *binary*, *short* and *long* clauses respectively
- fast restarts initially; restarts slow down later

$$\exists\, a,b\ [\ \boxed{\forall x\ [\ \boxed{\exists\, c,d\ [\ f(a,b,c,d,x)\ ]}\ ]}\ ]$$

$$\equiv\quad \exists\, a,b\ [\ \boxed{\boxed{\exists\, c,d\ [\ f(a,b,c,d,x)\ ]}\ \langle x/1\rangle \wedge \boxed{\exists\, c,d\ [\ f(a,b,c,d,x)\ ]}\ \langle x/0\rangle}\ ]$$

$$\equiv\quad \exists\, a,b\ [\ \exists\, c,d,x\ [\ x \wedge f(a,b,c,d,x)\ ]\ \wedge\ \exists\ \boxed{c,d}\ [\ f(a,b,\boxed{c,d},0)\ ]\ ]$$

$$\equiv\quad \exists\, a,b\ [\ \exists\, c,d,x\ [\ x \wedge f(a,b,c,d,x)\ ]\ \wedge\ \exists\ \boxed{c',d'}\ [\ f(a,b,\boxed{c',d'},0)\ ]\ ]$$

$$\equiv\quad \exists\, a,b,c,d,c',d'\ [\ x \wedge f(a,b,c,d,x) \wedge f(a,b,c',d',0)\ ]$$

Elimination of Universal Quantifiers by Copying Clauses

$$\exists\, a,b\,[\ \forall x,y\,[\ \exists\, c,d\,[\ f(a,b,c,d,x,y)\ ]\ ]\ ]$$

$$\equiv \quad \exists\, a,b\,[\,\forall y\,[\ \forall x\,[\ \exists\, c,d\,[\ f(a,b,c,d,x,y)\ ]\ ]\ ]\,]$$

$$\equiv \quad \exists\, a,b\,[\,\forall y\,[\ \exists\, c,d\,[\ f(a,b,c,d,x,y)\ ] <x/1> \wedge\ \exists\, c,d\,[\ f(a,b,c,d,x,y)\ ] <x/0>\ ]\,]$$

$$\equiv \quad \exists\, a,b\,[\,\forall y\,[\,\exists\, c,d,x\,[\,x \wedge f(a,b,c,d,x,y)\,]\ \wedge\ \exists\ c,d\ [\ f(a,b,\ c,d\ ,0,y)\,]\,]\,]$$

$$\equiv \quad \exists\, a,b\,[\,\forall y\,[\,\exists\, c,d,x\,[\,x \wedge f(a,b,c,d,x,y)\,]\ \wedge\ \exists\ c',d'\ [\ f(a,b,\ c',d'\ ,0,y)\,]\,]\,]$$

$$\equiv \quad \exists\, a,b\,[\,\forall y\,[\,\exists\, c,d,c',d'\,[\,x \wedge f(a,b,c,d,x,y) \wedge f(a,b,c',d',0,y)\,]\,]\,]$$

Features

- elimination of innermost universal variable with most occurrences (DLIS)
  - heuristically maximizes the number of removed clauses after expansion
- simplification of CNF matrix and quantifier prefix by
  - unit resolution
  - pruning of unates, zombies, and empty scopes
  - elimination of satisfied clauses
  - garbage collection of indices
- for existential problems use builtin DPLL style SAT solver

Future Work

- quantify out innermost existential variables by resolution (DP)
  - investigate when to eliminate universal or existential variables
- simplify CNF by subsumption tests
- apply look-forward strategies like learning
  - facts involving universal variables may lead to early conflicts
  - existential implications or equivalences may lead to elimination
- compare with other quantifier elimination algorithms
  - eg [PlaistedBiereZhu] or simply use BDDs