THE SAT SOLVER FRAMEWORK PRISS

Norbert Manthey and Robert Stelzman

Trento, 16.06.2012

We do have powerful SAT solvers

Modern SAT solvers

- Solve SAT efficiently
- Can be wrapped for other tasks
- Provide some API to the outside
- Are special purpose (they solve a CNF formula)
- . . .

We do have powerful SAT solvers

Modern SAT solvers

- Solve SAT efficiently
- Can be wrapped for other tasks
- Provide some API to the outside
- Are special purpose (they solve a CNF formula)
- ...

However, for research we need more.

Statements that have been made

There has been work on splitting, e.g.

- If we split on backdoor variables first, the solver runs faster

Statements that have been made

There has been work on splitting, e.g.

- If we split on backdoor variables first,
  the solver runs faster
- Branching on auxiliary variables slows the solver down

Statements that have been made

There has been work on splitting, e.g.

- If we split on backdoor variables first,
  the solver runs faster
- Branching on auxiliary variables slows the solver down
- Can we prove this easily with existing tools?

Statements that have been made

There has been work on splitting, e.g.

- If we split on backdoor variables first,
  the solver runs faster
- Branching on auxiliary variables slows the solver down
- Can we prove this easily with existing tools?

- Can we extend the results to parallel solvers?
- We try to provide a modular solver that provides a starting point.

Outline

Motivation

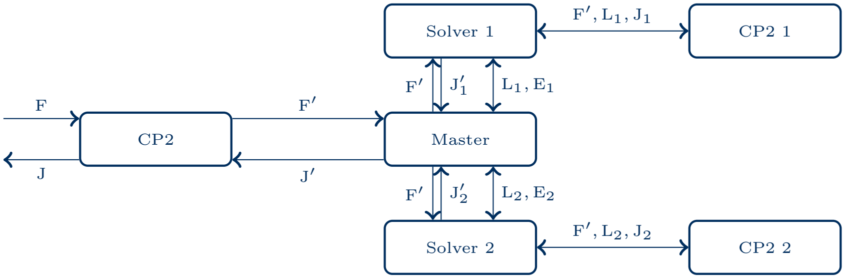
The parallel framework

- The SLS solver ssa
- The CDCL solver riss 2.0
- A special configuration

Demo

Conclusion

The solver (p)riss



The solver (p)riss

The framework currently implements two solvers that can be run in parallel

- The CDCL solver riss 2.0
- The SLS solve ssa

Clauses can be shared (and filtered) based on their

- Size, e.g. less than the average since the last restart
- LBD value
- PSM value

Furthermore,

- priss 2.0 shares equivalent literals
- ssa sends pure literals
- Sending and receiving can be disabled
- Messages from specific senders can be refused

Outline

Motivation

The parallel framework

- The SLS solver ssa
- The CDCL solver riss 2.0
- A special configuration

Demo

Conclusion

The SLS solver ssa

ssa is a re-implementation of sparrow 2011 with some modifications

- Broken clauses are weighted
- Variables have scores
- If we cannot do greedy steps, we do walksat

The SLS solver ssa

ssa is a re-implementation of sparrow 2011 with some modifications

- Broken clauses are weighted
- Variables have scores
- If we cannot do greedy steps, we do walksat

- We store promising variables in a heap
- We update variables in this heap incrementally

The SLS solver ssa

ssa is a re-implementation of sparrow 2011 with some modifications

- Broken clauses are weighted
- Variables have scores
- If we cannot do greedy steps, we do walksat

- We store promising variables in a heap
- We update variables in this heap incrementally
- We profiled the order of if-conditionds
- The solver is not tuned for a specific instance set

Outline

- Motivation
- The parallel framework
    - The SLS solver ssa
    - The CDCL solver riss 2.0
    - A special configuration
- Demo
- Conclusion

The CDCL solver riss 2.0

Initially, we wanted to implement MiniSAT in a modular way

- Each major component of the CDCL procedure can be replaced easily
- We want to exchange components easily

The CDCL solver riss 2.0

Initially, we wanted to implement MiniSAT in a modular way

- Each major component of the CDCL procedure can be replaced easily
- We want to exchange components easily
- We want to adapt the solver fast to research needs, e.g. student experiments
- We want to reproduce results from other publications

The CDCL solver riss 2.0

Initially, we wanted to implement MiniSAT in a modular way

- Each major component of the CDCL procedure can be replaced easily
- We want to exchange components easily
- We want to adapt the solver fast to research needs, e.g. student experiments
- We want to reproduce results from other publications

Some features that have been requested:

- Branch on a set of variables first
- Branch on a given polarity first
- Dump and load learned clauses
- Enumerate multiple models

Features that are provided

In riss 2.0, we provide:

- A C++ implementation with templates and library loaders
- All the above requested features
- A modular preprocessor (presentation on Sunday)
- Simplification during search (all techniques handle learned clauses)
- Several restart schemes (geometric, inner-outer, luby, dynamic)
- Removal heuristics (activity, LBD, PSM) and schedules
- Decision heuristics (VSIDS, VMTF), pick dominator
- Restricted extended resolution during learning
- Assignment shrinking
- Reused-Trail restart simulation
- ...

- Some analysis tools

Combining CDCL and SLS for parallel search

Based on ppfolio, and since we have a CDCL and an SLS solver

- Build a hybrid solver
- As in portfolio solvers, we want to share data

Combining CDCL and SLS for parallel search

Based on ppfolio, and since we have a CDCL and an SLS solver

- Build a hybrid solver
- As in portfolio solvers, we want to share data

What can we share?

- Learned clauses from CDCL to SLS
- Currently, pure literals from SLS to CDCL
- There is more to come ...

Outline

Motivation

The parallel framework

- The SLS solver ssa
- The CDCL solver riss 2.0
- A special configuration

Demo

Conclusion

Demonstrating the SLS solver

ssa can solve random SLS instances in Dimacs format

- After a preprocessor has been run
- Run alone
- . . .

Demonstrating the CDCL solver

riss 2.0 can

- Display its help (huge)

Demonstrating the CDCL solver

riss 2.0 can

- Display its help (huge)
- Solve CNF and output statistics
- Can share information with filters
- Be verbose
- Load components from libraries
- . . . and more

Note: the solver is not tuned on instance sets

Outline

Motivation

The parallel framework

- The SLS solver ssa
- The CDCL solver riss 2.0
- A special configuration

Demo

Conclusion

Future Work

What we need to do next:

- Improving the SLS solver
- Incorporate new techniques and more features into CDCL
- Tune the systems
  - for competition instances
  - for special applications
  - as parallel portfolio solver (diverse configurations)

Conclusion

priss 2.0 provides

- a modular, easy to extend CDCL solver
- a comparable SLS solver
- many features
- many parameters
- the ability to execute solvers in parallel with sharing information

Where can you find the tool?

`http://tools.computational-logic.org`

We provide

- Statically linked binaries
- The source code of the current version (under GPL 2)

- We will put updates and fixes online

Thanks for your attention

The solver is available at `http://tools.computational-logic.org`