Practical Course: SMT Solving

Introductory Meeting

Erika Ábrahám, Florian Corzilius, Gereon Kremer

Winter term 2015/2016

Satisfiability Checking

Problem definition: SAT

Given a logical formula $\varphi(x_1, ..., x_n)$ over boolean variables, is there an assignment for $x_1, ..., x_n$ such that $\varphi(x_1, ..., x_n)$ holds?

Satisfiability Checking

Problem definition: SAT

Given a logical formula $\varphi(x_1, ..., x_n)$ over boolean variables, is there an assignment for $x_1, ..., x_n$ such that $\varphi(x_1, ..., x_n)$ holds?

SAT is NP-complete, but can often be solved quickly.

Satisfiability Checking

Problem definition: SAT

Given a logical formula $\varphi(x_1, ..., x_n)$ over boolean variables, is there an assignment for $x_1, ..., x_n$ such that $\varphi(x_1, ..., x_n)$ holds?

SAT is NP-complete, but can often be solved quickly.

Ideas of DPLL SAT-solving

`Decide` an assignment for some variable
`Propagate` this assignment (check for deductions)
If `conflict` undo the assignments, `decide` differently

Satisfiability Checking

Problem definition: SAT

Given a logical formula $\varphi(x_1, ..., x_n)$ over boolean variables, is there an assignment for $x_1, ..., x_n$ such that $\varphi(x_1, ..., x_n)$ holds?

SAT is NP-complete, but can often be solved quickly.

Ideas of DPLL SAT-solving

`Decide` an assignment for some variable
`Propagate` this assignment (check for deductions)
If `conflict` undo the assignments, `decide` differently

Example

$\varphi(x, y, z) := x \vee ((y \wedge \neg z) \rightarrow x)$

Satisfiability Checking – SMT

Problem definition: Satisfiability Modulo Theories

Given a logical formula $\varphi(x_1, ..., x_n)$ over variables of some domain, is there an assignment for $x_1, ..., x_n$ such that $\varphi(x_1, ..., x_n)$ holds?

Satisfiability Checking – SMT

Problem definition: Satisfiability Modulo Theories

Given a logical formula $\varphi(x_1, ..., x_n)$ over variables of some domain, is there an assignment for $x_1, ..., x_n$ such that $\varphi(x_1, ..., x_n)$ holds?

The logical formula may contain atoms from some theory:

- Equalities over real or integer variables
- Equalities over Bitvector variables
- **Equalities over uninterpreted domains**

Satisfiability Checking – SMT

Problem definition: Satisfiability Modulo Theories

Given a logical formula $\varphi(x_1, ..., x_n)$ over variables of some domain, is there an assignment for $x_1, ..., x_n$ such that $\varphi(x_1, ..., x_n)$ holds?

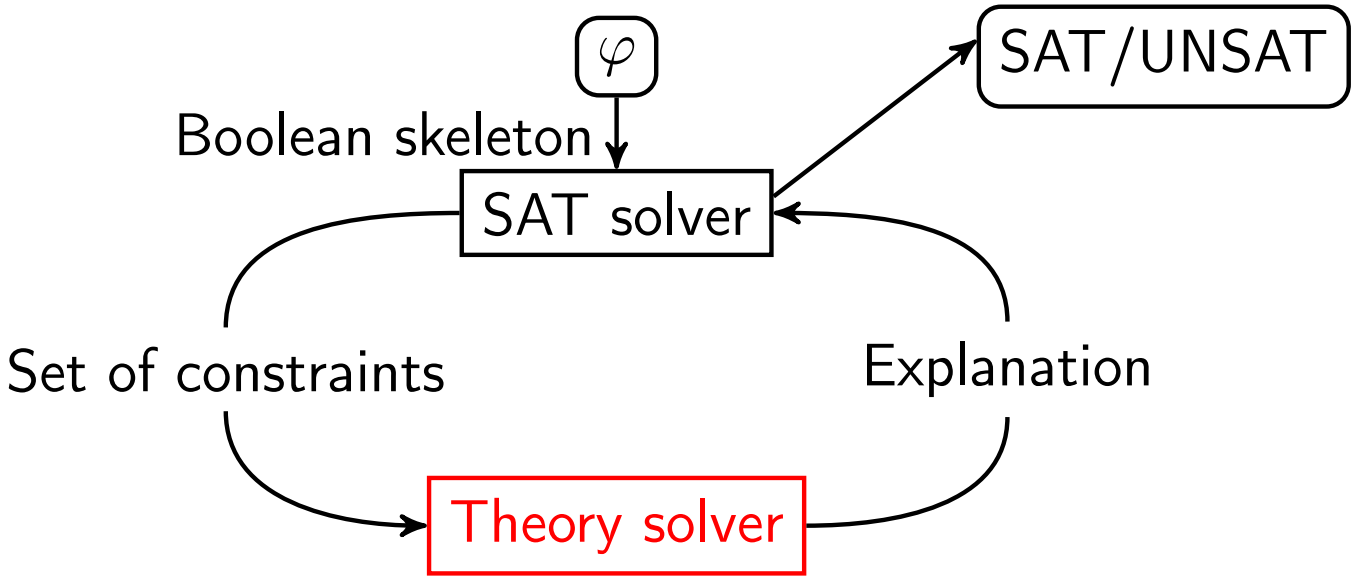
The logical formula may contain atoms from some theory:

- Equalities over real or integer variables
- Equalities over Bitvector variables
- **Equalities over uninterpreted domains**

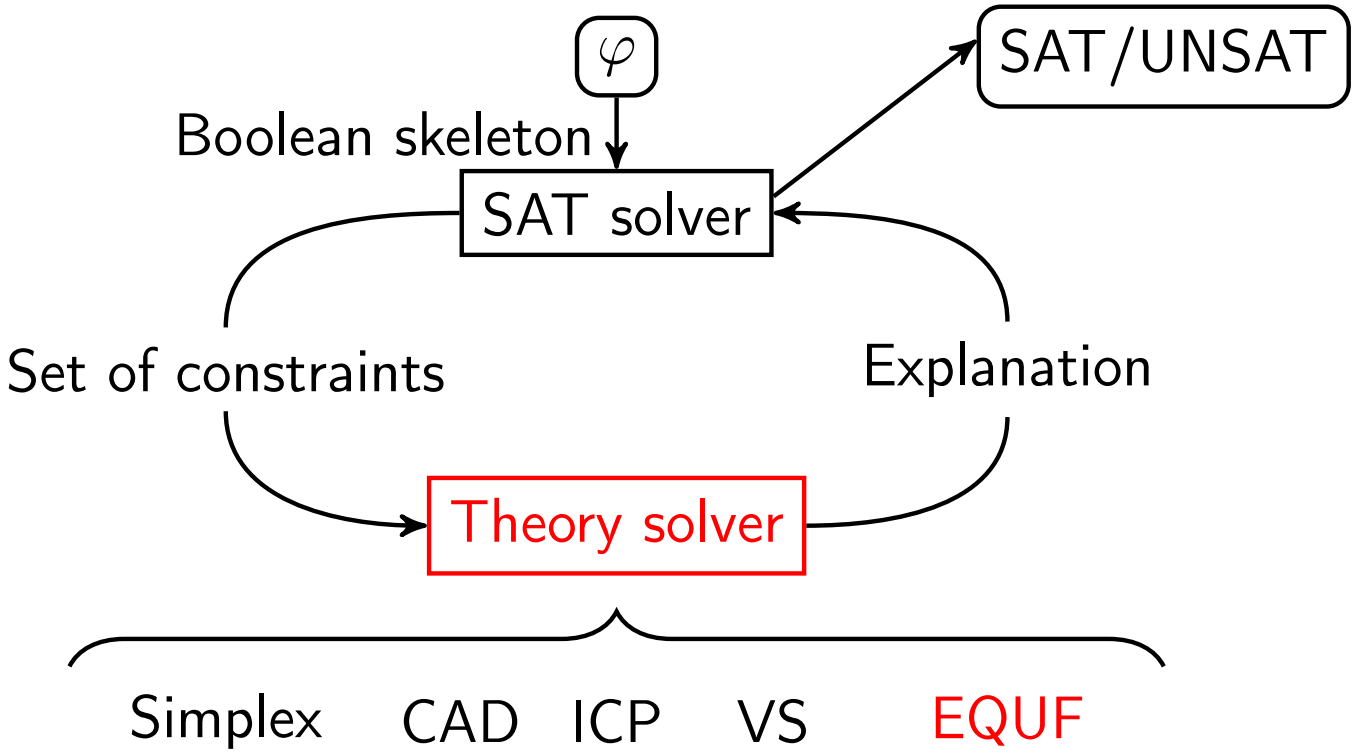
Examples

$\varphi(b, x, y, z) := (b \rightarrow x \cdot y \geq z) \wedge (\neg b \rightarrow y + x + z \leq 0), b \in \mathbb{B}, x, y, z \in \mathbb{R}$
$\varphi(a, b, c, d) := a = b \wedge b = c \wedge c = d \wedge a \neq d, a, b, c, d \in D$

SMT Solving


φ
Boolean skeleton
SAT/UNSAT
SAT solver
Set of constraints
Explanation
Theory solver

SMT Solving



Theory solver

Gets a set of constraints
Decides whether the constraints are consistent
Returns `SAT` or `UNSAT` with an explanation

Goals of this practical course

- Understanding of SMT solving
- Understanding of theories: `QF_UF`, `QF_NRA`, `QF_UFNRA`, ...
- Understanding of different decision procedures for equality logic and uninterpreted functions

Goals of this practical course

- Understanding of SMT solving
- Understanding of theories: `QF_UF`, `QF_NRA`, `QF_UFNRA`, ...
- Understanding of different decision procedures for equality logic and uninterpreted functions
- Implementation of these procedures as theory modules in SMT-RAT
- Implementation in clean and modern C++
- Debugging, evaluation and documentation of theory modules
- Presentation of results

Setup

We have multiple teams ($X \in \{a, b, ...\}$)

- A mailinglist `smt-X@ths.informatik.rwth-aachen.de`
- Read access to CArL and SMT-RAT repositories
- A git repository containing a clone of SMT-RAT:
  `https://srv-i2.informatik.rwth-aachen.de/scm/git/smtws15/smt-X.git`

Setup

We have multiple teams ($X \in \{a, b, ...\}$)

- A mailinglist `smt-X@ths.informatik.rwth-aachen.de`
- Read access to CArL and SMT-RAT repositories
- A git repository containing a clone of SMT-RAT:
  `https://srv-i2.informatik.rwth-aachen.de/scm/git/smtws15/smt-X.git`
- Anything else? Trac? Wiki?

Setup

We have multiple teams ($X \in \{a, b, ...\}$)

- A mailinglist `smt-X@ths.informatik.rwth-aachen.de`
- Read access to CArL and SMT-RAT repositories
- A git repository containing a clone of SMT-RAT:
  `https://srv-i2.informatik.rwth-aachen.de/scm/git/smtws15/smt-X.git`
- Anything else? Trac? Wiki?
- You need: Linux or MacOS with the following software:
  git, cmake, ccmake, cln, gmp, eigen3, g++ ($\geq$ 4.8) or clang ($\geq$ 3.4), boost, doxygen, gtest

Setup

We have multiple teams ($X \in \{a, b, ...\}$)

- A mailinglist `smt-X@ths.informatik.rwth-aachen.de`
- Read access to CArL and SMT-RAT repositories
- A git repository containing a clone of SMT-RAT:
  `https://srv-i2.informatik.rwth-aachen.de/scm/git/smtws15/smt-X.git`
- Anything else? Trac? Wiki?
- You need: Linux or MacOS with the following software:
  git, cmake, ccmake, cln, gmp, eigen3, g++ ($\geq$ 4.8) or clang ($\geq$ 3.4), boost, doxygen, gtest

Changes to CArL or the core of SMT-RAT will be committed by us and available to both teams

Roadmap

- Design an algorithm for equality logic and uninterpreted functions
- Design datastructures supporting this algorithm
- Presentation of design: October / November
- Implementation as a theory module
- Compare different heuristics and optimizations
- Test on standard benchmarks
- Presentation of results: January / February

Building groups

Meetings

Weekly:

- Meeting in the seminar room
- Not mandatory, but encouraged
- You can discuss, ask for help, work/implement, ...

Meetings

Weekly:

- Meeting in the seminar room
- Not mandatory, but encouraged
- You can discuss, ask for help, work/implement, ...

Monthly (roughly every fourth meeting):

- Mandatory
- Discussion of results
- Presentation of new tasks

URLs

- Homepage:
  `http://ths.rwth-aachen.de/teaching/ws15/swp-smt-solving/`
- Supervisors: `smt-orga@ths.informatik.rwth-aachen.de`
- Everyone: `smt@ths.informatik.rwth-aachen.de`
- Your team: `smt-X@ths.informatik.rwth-aachen.de`
- CArL:
  `https://<user>@srv-i2.informatik.rwth-aachen.de/scm/git/carl.git`
- SMT-RAT:
  `https://<user>@srv-i2.informatik.rwth-aachen.de/scm/git/smtrat.git`
- Your git: `https://<user>@srv-i2.informatik.rwth-aachen.de/scm/git/smtws15/smt-X.git`
- Documentation for CArL (includes introduction to our build process):
  `https://smtrat.github.io/carl/`

References

Roberto Bruttomesso, Alessandro Cimatti, Anders Franzén, Alberto Griggio, Alessandro Santuari, and Roberto Sebastiani.

To Ackermann-ize or not to Ackermann-ize? On Efficiently Handling Uninterpreted Function Symbols in SMT (EUF $\bigcup$ T).

In *LPAR*, pages 557–571. Springer, 2006.

Florian Corzilius, Ulrich Loup, Sebastian Junges, and Erika Ábrahám.

SMT-RAT: An SMT-Compliant Nonlinear Real Arithmetic Toolbox.

In *Theory and Applications of Satisfiability Testing*, LNCS, pages 442–448. Springer, 2012.

Daniel Kroening and Ofer Strichman.

Decision Procedures: An Algorithmic Point of View, pages 59–110.

Springer, 2008.

That's it...

Questions?