Automated Deduction

Laura Kovács
TU Wien

Outline

Combinations of Theories

Combination of theories: Models of Satisfiable Sets

Running an SMT Solver

First-Order Theorem Proving - An Example

Example (recap)

$$x + 2 = y$$
$$f(read(write(a, x, 3), y - 2)) \neq f(y - x + 1)$$

For deciding satisfiabiliy:

- Separate reasoning in various theories

  - introduce new variables that are shared by all theories

- Make reasoners exchange equalities

  - derive and use equalities among shared variables

Separating Reasoning

$$x + 2 = y$$
$$f(read(write(a, x, 3), y - 2)) \neq f(y - x + 1)$$

Linear Arithmetic	Uninterpreted Functions
$y = x + 2$	$f(c_4) \neq f(c_1)$
$c_1 = y - x + 1$	
$c_2 = y - 2$	
$c_3 = 3$	

Arrays

$$c_4 = read(write(a, x, c_3), c_2)$$

- Each step obviously preserves satisfiability; moreover every model of the modified set of literals is also a model of the original set.
- In this example every "separated" subset of literals is satisfiable.
- But this does not mean that the set of all literals is satisfiable.

Exchanging Equalities

Linear Arithmetic	Uninterpreted Functions
$y = x + 2$ $c_1 = y - x + 1$ $c_2 = y - 2$ $c_3 = 3$	$f(c_4) \neq f(c_1)$
Arrays	**Equalities**
$c_4 = read(write(a, x, c_3), c_2)$	$c_2 = x$ $c_4 = c_3$ $c_1 = c_3$

- Congruence closure returns "unsatisfiable".
- All derived equalities are implied by the initial set of literals. Therefore, the initial set is unsatisfiable.
- Therefore the original set (without extra variables) is unsatisfiable, too.

Not all theories can be combined (easily) in this way!

Not all theories can be combined in this way!

Restrictions on theories $\mathcal{T}_1, \ldots, \mathcal{T}_n$:

1. Theories have disjoint signatures;
2. Each theory $\mathcal{T}_i$ is stably infinite;
   - A theory $\mathcal{T}_i$ with signature Σ_i is stably infinite if for every satisfiable formula $F \in \mathcal{T}_i$ there exists some $\mathcal{T}$-interpretation such that:
     - $I \models F$, and
     - I has a domain of infinite cardinality.

$\mathcal{T}_E, \mathcal{T}_A, \mathcal{T}_Q$ are stably infinite.

Problem with combining theories, as described, that are not stably infinite. Why?

Not all theories can be combined EASILY in this way!

Restrictions on theories $\mathcal{T}_1, \ldots, \mathcal{T}_n$:

1. Theories have disjoint signatures;
2. Each theory $\mathcal{T}_i$ is stably infinite;
3. Each theory $\mathcal{T}_i$ is convex.
   - A theory $\mathcal{T}_i$ is convex if for every formula $F \in \mathcal{T}_i$ such that F is a conjunction of $\mathcal{T}_i$-literals:

$$\text{if } F \to \bigvee_{j=1}^{k} (u_j = v_j) \quad \text{then} \quad F \to u_j = v_j \text{ for some } j \in \{1, \ldots, k\}.$$

Is $\mathcal{T}_{\mathbb{Z}}$ convex? No.

Is $\mathcal{T}_A$ convex? No.

$\mathcal{T}_E, \mathcal{T}_Q$ are convex.

Not all theories can be combined EASILY in this way!

Restrictions on theories $\mathcal{T}_1, \ldots, \mathcal{T}_n$:

1. Theories have disjoint signatures;
2. Each theory $\mathcal{T}_i$ is stably infinite;
3. Each theory $\mathcal{T}_i$ is convex.

Exchange of equalities between $\mathcal{T}_1, \ldots, \mathcal{T}_n$:

- for a convex theory $\mathcal{T}_i$, its decision procedure discovers a new equality $u = v$, for shared u, v; Pass this new equality to the decision procedures of the other theories $\mathcal{T}_j$;
- for a non-convex theory $\mathcal{T}_i$, its decision procedure discovers a disjunction of new equalities $\bigvee_k u_k = v_k$, for shared u_k, v_k;
  - Split the disjunction and exchange equalities along multiple branches: for each $u_k = v_k$, pass the equality to the decision procedures of the other theories $\mathcal{T}_j$.

    For efficiency, work with minimal disjunctions.

Combination of theories: The Nelson-Oppen method

The Nelson-Oppen method (1979):

- for reasoning in combination of stably infinite theories with disjoint signatures;
- by separating reasoning and exchange of equalities;
- for convex theories $\mathcal{T}_i$ with decision procedures in P, the Nelson-Oppen method is in P;
- for theories $\mathcal{T}_i$ with decision procedures in NP, the Nelson-Oppen method is in NP.

Outline

Combinations of Theories

Combination of theories: Models of Satisfiable Sets

Running an SMT Solver

First-Order Theorem Proving - An Example

Models of Satisfiable Sets

Linear Arithmetic	Uninterpreted Functions
$y = x + 2$ $c_1 = y - x + 2$ $c_2 = y - 2$ $c_3 = 3$	$f(c_4) \neq f(c_1)$
Arrays $c_4 = read(write(a, x, c_3), c_2)$	Equalities $c_2 = x$ $c_4 = c_3$ $c_4 = 3$ $c_1 = 4$

Take models of:

- the literals in every theory plus the set of derived equalities.

Outline

Combinations of Theories

Combination of theories: Models of Satisfiable Sets

Running an SMT Solver

First-Order Theorem Proving - An Example

Running an SMT Solver

The most efficient SMT solver: Z3, `http://rise4fun.com/Z3`.

SMTLib2 input format.

Easy to use.

SMT Related Problems – Where are we now?

- ✓ Deciding theory: Check satisfiability of a set of *literals* in $\mathcal{T}_E$, $\mathcal{T}_A$
- ✓ SMT and non-unit formulas: Put together theory reasoning and SAT solving
- ✓ Reasoning in combination of theories: Given decision procedures for theories, we can build a decision procedure for the combination of these theories.

Outline

Combinations of Theories

Combination of theories: Models of Satisfiable Sets

Running an SMT Solver

First-Order Theorem Proving - An Example

First-Order Theorem Proving

We will use the VAMPIRE theorem prover throughout the lecture.

Go to

`https://vprover.github.io/download.html`

and pick the route most suitable to you.

Notes:

- For Linux users, a binary is probably the easiest route
- For Mac users, you need to build from source
  - run `make vampire_rel`
- For Windows users, the easiest route for this tutorial is a virtual machine and then use Linux

First-Order Theorem Proving. An Example

Group theory theorem: if a group satisfies the identity $x^2 = 1$, then it is commutative.
More formally: in a group "assuming that $x^2 = 1$ for all x prove that $x \cdot y = y \cdot x$ holds for all x, y."
What is implicit: axioms of the group theory.

$$\forall x(1 \cdot x = x)$$
$$\forall x(x^{-1} \cdot x = 1)$$
$$\forall x \forall y \forall z((x \cdot y) \cdot z = x \cdot (y \cdot z))$$

Formulation in First-Order Logic

Axioms (of group theory):	$\forall x(1 \cdot x = x)$
	$\forall x(x^{-1} \cdot x = 1)$
	$\forall x \forall y \forall z((x \cdot y) \cdot z = x \cdot (y \cdot z))$
Assumptions:	$\forall x(x \cdot x = 1)$
Conjecture:	$\forall x \forall y(x \cdot y = y \cdot x)$

In the TPTP Syntax

The TPTP library (Thousands of Problems for Theorem Provers), http://www.tptp.org contains a large collection of first-order problems.
For representing these problems it uses the TPTP syntax, which is understood by all modern theorem provers, including Vampire.
In the TPTP syntax this group theory problem can be written down as follows:

```
%---- 1 * x = x
fof(left_identity,axiom,
    ! [X] : mult(e,X) = X).
%---- i(x) * x = 1
fof(left_inverse,axiom,
    ! [X] : mult(inverse(X),X) = e).
%---- (x * y) * z = x * (y * z)
fof(associativity,axiom,
    ! [X,Y,Z] : mult(mult(X,Y),Z) = mult(X,mult(Y,Z))).
%---- x * x = 1
fof(group_of_order_2,hypothesis,
    ! [X] : mult(X,X) = e).
%---- prove x * y = y * x
fof(commutativity,conjecture,
    ! [X] : mult(X,Y) = mult(Y,X)).
```

Running Vampire on a TPTP file

is easy: simply use

```
vampire <filename>
```

One can also run Vampire with various options, some of them will be explained later. For example, save the group theory problem in a file `group.tptp` and try

```
vampire --thanks TUWien group.tptp
```

Proof by Vampire (Slightliy Modified)

```
Refutation found.
270. $false [trivial inequality removal 269]
269. mult(sk0,sk1) != mult (sk0,sk1) [superposition 14,125]
125. mult(X2,X3) = mult(X3,X2) [superposition 21,90]
90. mult(X4,mult(X3,X4)) = X3 [forward demodulation 75,27]
75. mult(inverse(X3),e) = mult(X4,mult(X3,X4)) [superposition 22,19]
27. mult(inverse(X2),e) = X2 [superposition 21,11]
22. mult(inverse(X4),mult(X4,X5)) = X5 [forward demodulation 17,10]
21. mult(X0,mult(X0,X1)) = X1 [forward demodulation 15,10]
19. e = mult(X0,mult(X1,mult(X0,X1))) [superposition 12,13]
17. mult(e,X5) = mult(inverse(X4),mult(X4,X5)) [superposition 12,11]
15. mult(e,X1) = mult(X0,mult(X0,X1)) [superposition 12,13]
14. mult(sK0,sK1) != mult(sK1,sK0) [cnf transformation 9]
13. e = mult(X0,X0) [cnf transformation 4]
12. mult(X0,mult(X1,X2)) = mult(mult(X0,X1),X2) [cnf transformation 3]
11. e = mult(inverse(X0),X0) [cnf transformation 2]
10. mult(e,X0) = X0 [cnf transformation 1]
9. mult(sK0,sK1) != mult(sK1,sK0) [skolemisation 7,8]
8. ?[X0,X1]: mult(X0,X1) != mult(X1,X0) <=> mult(sK0,sK1) != mult(sK1,sK0)
                                                    [choice axiom]
7. ?[X0,X1]: mult(X0,X1) != mult(X1,X0) [ennf transformation 6]
6. ~![X0,X1]: mult(X0,X1) = mult(X1,X0) [negated conjecture 5]
5. ![X0,X1]: mult(X0,X1) = mult(X1,X0) [input]
4. ![X0]: e = mult(X0,X0)[input]
3. ![X0,X1,X2]: mult(X0,mult(X1,X2)) = mult(mult(X0,X1),X2) [input]
2. ![X0]: e = mult(inverse(X0),X0) [input]
1. ![X0]: mult(e,X0) = X0 [input]
```

- Each inference derives a formula from zero or more other formulas;
- Input, preprocessing, new symbols introduction, superposition calculus
- Proof by refutation, generating and simplifying inferences, unused formulas . . .