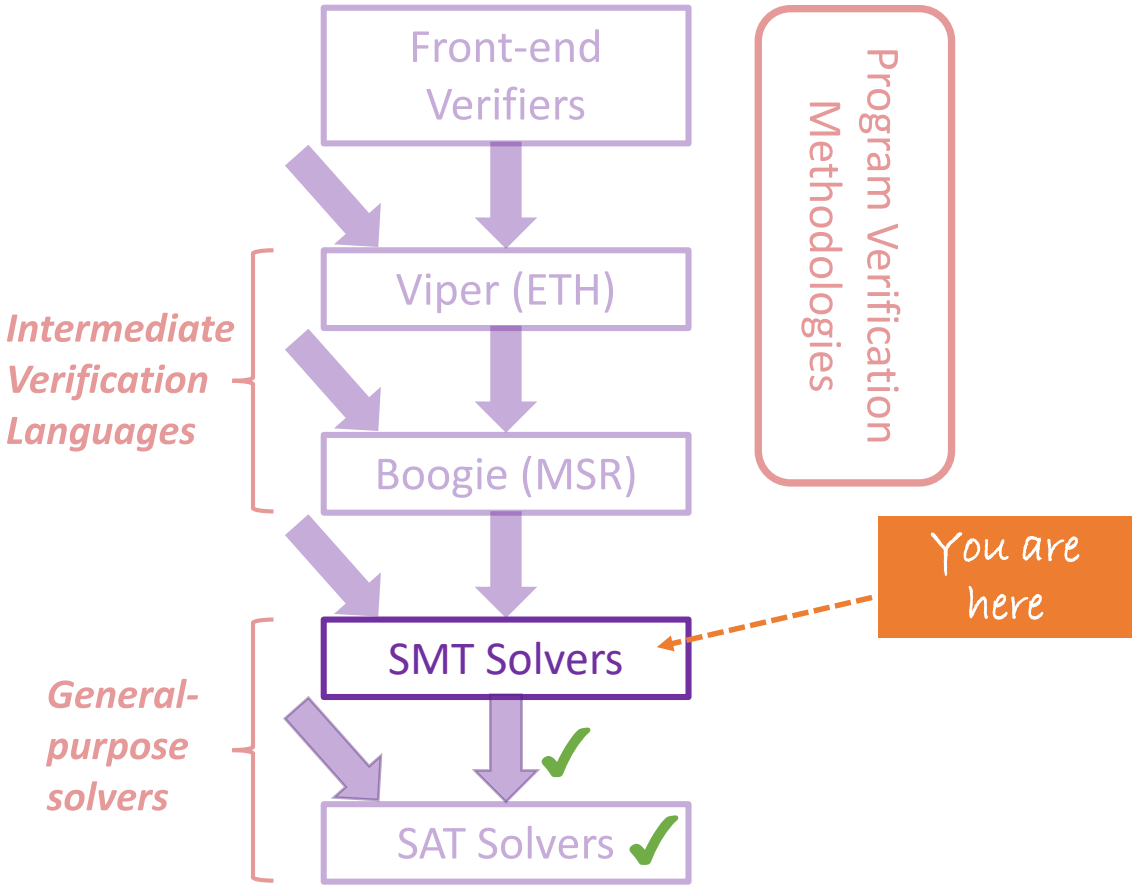
3. SMT Solving Algorithms

Program Verification

ETH Zurich, Spring Semester 2017

Alexander J. Summers

Next up: SMT Solving Algorithms



Satisfiability Modulo Theories

- The **SMT problem**:

  Given a first-order logic formula, with interpreted/uninterpreted symbols from (possibly several) theories, does it have a model?

- i.e. is the formula *satisfiable*? If so, *how*?
- Theories require custom reasoning engines: *theory solvers*
  - e.g. integer arithmetic, set theory, ADTs, reals, floating points, bit-vectors
  - is $\neg(f(i)=f(j)) \wedge i\text{-}j=0$ satisfiable? (*theory combination*)
- The *SAT problem* is a special case, in which:
  - the input formula is quantifier-free, without function symbols or equality
  - no theories are used
- SAT solving algorithms are an *important ingredient* in *SMT solvers*

Theories

- Recall: a *signature* is a set of sorts and function symbols
- From now on, we assume signatures to include a *standard elements*:
  - (interpreted) equality $=$ over terms from each sort
  - the (interpreted) Bool sort (along with $\top$, $\bot$, etc.)
- A *theory* T is a pair of a signature $\mathrm{sig}(\mathrm{T})$ and set of models $\mathrm{models}(\mathrm{T})$
  - the domain of all models in $\mathrm{models}(\mathrm{T})$ must be $\mathrm{sig}(\mathrm{T})$
  - all elements of $\mathrm{models}(\mathrm{T})$ must interpret standard symbols as usual
  - we typically omit standard elements, when concretely describing $\mathrm{sig}(\mathrm{T})$
- An alternative definition of theories is to define a theory via a *set of closed formulas* (those with no free variables – also called *sentences*)
  - $\mathrm{sig}(\mathrm{T})$ is the set of sort and function symbols used in the formulas
  - $\mathrm{models}(\mathrm{T})$ is the set of models in which the formulas are true

Examples of Theories

- The theory of *Equality and Uninterpreted Functions* $\mathrm{T_E}$ has a (any) signature consisting of only uninterpreted function symbols and sorts
  - the set $\mathrm{models}(\mathrm{T})$ is the set of all models for these functions and sorts
- The theory of *Presburger Arithmetic* $\mathrm{T_P}$ has signature $\{\mathrm{Int}, 0, 1, +, -, \leq\}$ with the usual interpretation (defining $\mathrm{models}(\mathrm{T})$)
- The theory of *Non-linear Arithmetic* $\mathrm{T_Z}$ has signature $\{\mathrm{Int}, 0, 1, +, -, \cdot, /, \leq\}$ with the usual interpretation
- The theory of *Real Arithmetic* $\mathrm{T_R}$ has signature $\{\mathrm{Real}, 0, 1, +, -, \cdot, /, \leq\}$ with the usual interpretation
- Overlaps between theory signatures may or may not be intended
  - if not, one can rename symbols to avoid clashes

Theory-Related Definitions

- For a signature S, a formula A is an S–*formula* iff A contains only sort and function symbols belonging to S
- A formula A is a T–*formula* iff A is a $\mathrm{sig}(\mathrm{T})$-formula
- A formula A is T–*satisfiable* iff there exists $\mathrm{M} \in \mathrm{models}(\mathrm{T})$ s.t. $\mathrm{M} \models \mathrm{A}$
  - note: this implicitly requires that A is a T-formula
- A set of T-formulas Γ T–*entails* a T-formula A, written $\Gamma \models_{\mathrm{T}} \mathrm{A}$, iff for all $\mathrm{M} \in \mathrm{models}(\mathrm{T})$: if M satisfies all formulas in Γ, then $\mathrm{M} \models \mathrm{A}$
- A set of T-formulas Γ is T–*consistent* iff $\Gamma \not\models_{\mathrm{T}} \bot$
- Two theories T_1 and T_2 are *disjoint* if their signatures overlap *only on standard elements, uninterpreted constants and uninterpreted sorts*
  - e.g. T_{E} and T_{Z} are disjoint theories, T_{P} and T_{Z} are not

First-order Literals and Clauses

- A *atom* is a formula without propositional connectives or quantifiers
  - depending on the signature, $\mathrm{f(a)=b}$, $\mathrm{m \cdot n \leq 42}$ could be atoms; 42 is not
  - a *propositional atom* is an uninterpreted constant symbol of sort Bool
- A (first-order) *literal* is an atom or its negation
  - For a literal l we write $\sim\mathrm{l}$ for the negation of l cancelling double negations
- A *clause* is a disjunction of (any finite number of) literals
- A T*–atom* is a T-formula which is also an atom (similarly for S*–atom*)
- A T*–literal* is a T-formula which is also a literal (similarly for S*–literal*)
- Conjunctive Normal Form is defined as previously
  - (a conjunction of disjunctions of literals)
  - Note: formulas in CNF are quantifier-free

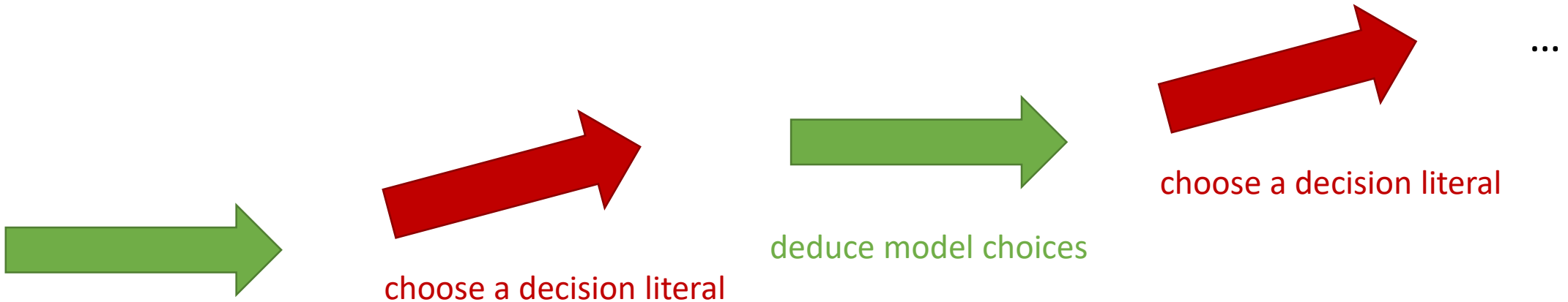
Propositional Abstraction

- For a given signature S, we define a signature S^{p} containing (only):
  - the propositional S-atoms
  - a *fresh propositional atom* for each non-propositional S-atom
- We then fix a *bijective mapping* from the non-propositional S-atoms to the set of S^{p}-atoms
- For an S-formula A, the formula A^{p} is the *propositional abstraction* of A, given by *replacing* all non-propositional atoms in A with their image under this mapping
- An S-formula A is *propositionally unsatisfiable* if $\mathrm{A}^{\mathrm{p}} \models \bot$
- An S-formula A *propositionally entails* an S-formula B iff $\mathrm{A}^{\mathrm{p}} \models \mathrm{B}^{\mathrm{p}}$
  - Note: $\mathrm{A}^{\mathrm{p}} \models \mathrm{B}^{\mathrm{p}}$ implies $\mathrm{A} \models \mathrm{B}$ but *not necessarily vice versa*

Theory Solvers

- A theory solver can answer (ideally all of) the following questions:
- Is a given set of T-literals Γ T–consistent?
  - if so, what is a T-model satisfying them?
  - if not, what is a (preferably minimal) T–inconsistent subset of Γ? (*conflict set*)
- For a T–consistent set Γ of T-literals, are there extra implied literals?
  - If so, what are they? (e.g. return l such that $\Gamma \models_{\mathrm{T}} \mathrm{l}$)
  - Typically, these answers are restricted to literals in the input formula (why?)
- For a T–consistent set Γ, are there any implied *interface equalities*?
  - ideally, return $\mathrm{e}_1 = \mathrm{e}_2$ such that $\Gamma \models_{\mathrm{T}} \mathrm{e}_1 = \mathrm{e}_2$ and $\mathrm{e}_1, \mathrm{e}_2$ are terms in Γ
  - or, return an implied disjunction of such equalities
  - The terminology “interface equalities” comes from their importance for sharing information during *theory combination* (see later)

Recall: DPLL / CDCL Algorithms



(simplification steps, pure literal rule, unit clause propagation)

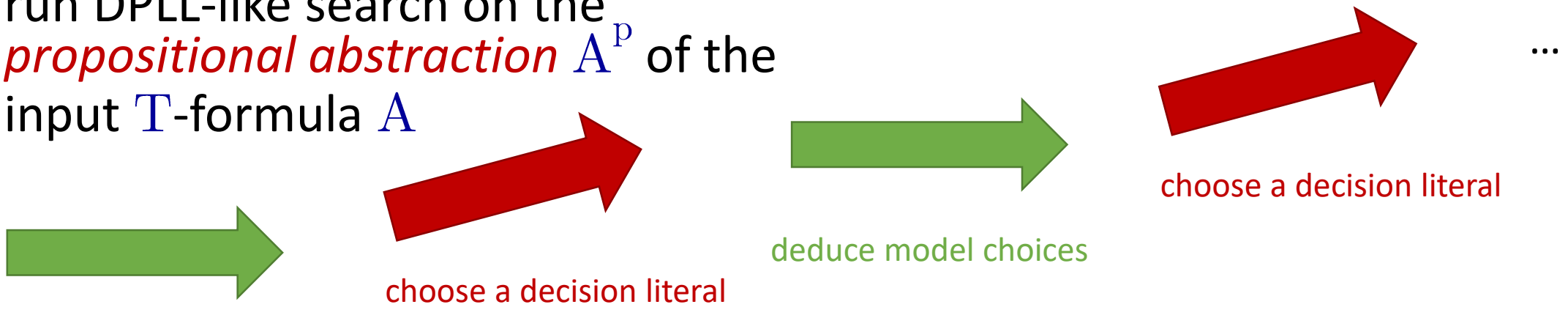
(may need to backtrack)

…until…

- conflict reached:
  - backtrack – try flipping a decision literal
  - (if CDCL) learn new clause, back-jump
- model found:
  - return the model

Adapting DPLL to DPLL(T)

run DPLL-like search on the *propositional abstraction* A^{p} of the input T-formula A



…until…

- conflict reached: backtrack/jump, learn clauses as usual
- model found (represented by set of literals Γ):
  - is it a T-model? *Not necessarily (why not?)*
  - Ask theory solver: is Γ is T–consistent? If yes, we're done
  - If no, backtrack in the original search
  - (CDCL) get a T–inconsistent subset for clause learning / back-jumping

Extended Theory Solver Integration

- The described adaptation to DPLL/CDCL works for a single theory
- There are several further possibilities, even in the single-theory case
- During deduction of model choices, we can also ask theory solver:
  - given current candidate model Γ, are there extra implied literals?
  - If so, we add to the model all l such that $\Gamma \models_T l$ (*theory deduction*)
  - We can detect inconsistency (*early pruning*) by checking T–consistency of Γ
- In CDCL, we can learn not only from propositional conflicts
  - by obtaining a T–inconsistent set Γ' (i.e. $\Gamma' \models_T \bot$) we can:
  - generate a *theory conflict clause* $\bigvee_{l \in \Gamma'} \sim l$ (capturing the theory conflict)
  - alternatively, use a "cut" in the implication graph before these nodes
  - we then obtain a *mixed propositional/theory conflict clause*
  - back-jumping is also determined by the decision literals relevant for Γ'

Theory Combination

- In practice, we encounter problems which require *multiple theories*
  - program verification: theories for program datatypes, to encode memory, etc.
- We will now turn to combining two theory solvers in one problem
  - the same approach then generalises to any finite number of solvers
- We focus here on the case of combining solvers for *disjoint theories*
  - extensions are possible, and build upon these ideas
- The approach described here is called a *lazy SMT* approach
  - lazy means we don't try to decide theory-related aspects up-front (eagerly)
  - *eager SMT* eliminates (some) theories: e.g. techniques from the last lecture

Two Theories – An Idea

- Suppose we have an input formula A which includes symbols (only) from *disjoint theories* T_1 and T_2 for which we have theory solvers
- Suppose that we can find A_1, A_2 and a set S of constant symbols s.t. :
  - $\mathrm{A} \equiv \mathrm{A}_1 \wedge \mathrm{A}_2$ (and all constant symbols in S are fresh)
  - A_1 contains only function symbols from T_1 and S
  - A_2 contains only function symbols from T_2 and S
- Now, we can ask each theory solver to check its respective conjunct
- If either returns `unsat` then we can terminate with this result (why?)
- If both return `sat` then it *could still be* that A is unsatisfiable
  - for example, consider $\neg(\mathrm{f}(\mathrm{i})=\mathrm{f}(\mathrm{j})) \wedge \mathrm{i}\text{-}\mathrm{j}=0$
  - the first conjunct is satisfiable, but only if $\neg(\mathrm{i}=\mathrm{j})$, the second is, but only if $\mathrm{i}=\mathrm{j}$
  - we need some way of sharing information. In general, *what information?*

Aside - Interpolation

- Craig's Interpolation Theorem (1957) says, for first-order A_1 and A_2 :
- If $\mathrm{A} \models \mathrm{B}$ then there exists C such that:
  - $\mathrm{A} \models \mathrm{C}$ and $\mathrm{C} \models \mathrm{B}$
  - the only function symbols in C are those occurring in *both* A and B
- Now, let A be A_1 and B be $\neg \mathrm{A}_2$ (from the previous slide)
- We have: $\mathrm{A}_1 \models \neg \mathrm{A}_2$ if and only if $\mathrm{A}_1 \wedge \mathrm{A}_2 \models \bot$
- Therefore $\mathrm{A}_1 \wedge \mathrm{A}_2$ *is unsatisfiable* if and only if there exists C s.t.
  - C has *no function symbols other than equalities and uninterpreted constants*
  - $\mathrm{A}_1 \wedge \neg \mathrm{C}$ is unsatisfiable and $\mathrm{A}_2 \wedge \mathrm{C}$ is unsatisfiable
- The two sub-problems (only) need to share (dis-)equality information
  - and this information only need involve the constants in both conjuncts

Purification

- In actual fact, we don't have to split the entire formula into parts
- It is sufficient for any *candidate model* to be splittable into two
- We just need each *literal* in the input formula to belong to one theory
- *Purification* achieves this (in a similar way to Tseitin's CNF conversion):
- For each literal l in A involving terms from both theories:
  - select a subterm t of l containing symbols from one theory but not the other
  - choose a *fresh constant symbol* c
  - rewrite l in A by replacing t with c
  - conjoin $\mathrm{c}{=}\mathrm{t}$ to A
  - repeat until all literals involve symbols from at most one theory
- After purification, the fresh constants chosen along with any originally in both theories, are called the *shared constants*

Non-Deterministic Nelson-Oppen Combination

- Suppose we have an input formula A from *disjoint, stably-infinite theories* (see next slide) T_1 and T_2 for which we have theory solvers

1. Purify the input formula A to B; let S be the shared constants
2. Pick any equivalence relation on S; according to it, conjoin either an equality or a disequality to B for each pair of elements in S
3. Use the theory extension of DPLL (slide 66), modified as follows:
4. When we have to check a candidate model using the theory solver:
   - query each theory solver but *filter-out literals from the candidate model* which include function symbols belonging only to the other theory
   - if both return `sat` we are done, otherwise process a conflict as usual
5. If the DPLL-search fails to find a model, return to step 2 and choose a new equivalence relation on S; if all have been tried return `unsat`

Stably-Infinite Theories

- A model $\mathrm{M'}$ is an *extension* of a model M, if:
  - for every uninterpreted sort $\mathrm{S} \in \mathrm{dom}(\mathrm{M})$, $\mathrm{S} \in \mathrm{dom}(\mathrm{M'})$ and $\mathrm{M}(\mathrm{S}) \subseteq \mathrm{M'}(\mathrm{S})$
  - for every term t: if $\ulcorner \mathrm{t} \urcorner_{\mathrm{M}}$ is defined then $\ulcorner \mathrm{t} \urcorner_{\mathrm{M'}}$ is defined and $\ulcorner \mathrm{t} \urcorner_{\mathrm{M'}} = \ulcorner \mathrm{t} \urcorner_{\mathrm{M}}$
- Intuitively, an extension of a model can enlarge the interpretation of uninterpreted sorts, and interpret more function symbols and sorts

- A theory T is *stably-infinite* if, for any $\mathrm{M} \in \mathrm{models}(\mathrm{T})$ and T-formula A such that $\mathrm{M} \models \mathrm{A}$, in any extension $\mathrm{M'}$ of M, we have $\mathrm{M'} \models \mathrm{A}$
- Informally, this requirement prevents a theory from imposing hard constraints on the size of models
  - for example, a theory cannot satisfy $\forall \mathrm{x}{:}\mathrm{T}.\ \mathrm{x} = \mathrm{c}$ for uninterpreted sort T
  - this requirement allows partial models from solvers to be safely combined

Convex Theories

- A theory T is *convex* if, for all finite sets of literals Γ and for all (non-empty) disjunctions of equalities $V_{i\in\{0,\ldots,n\}} x_i = y_i$:

  if $\Gamma \vDash_T V_{i\in\{0,\ldots,n\}} x_i = y_i$ then $\Gamma \vDash_T x_j = y_j$ for some $j \in \{0, \ldots, n\}$

- For convex theories, one can improve on the "guessing" of equalities
- Deterministic Nelson-Oppen (for two convex, stably-infinite theories):
  - omit steps 2 and 5 of the algorithm on slide 72
  - each theory solver can be made (efficiently) to enumerate and output *all* (dis-)equalities on shared constants implied by the current candidate model
  - these equalities can then be conjoined to the current model, and the other solver invoked again
  - this process repeats until one or other solver fails to find a new constraint
  - since there are finitely many shared constants, termination is guaranteed

Nelson-Oppen Combinations - Results

- Given an input formula A from *disjoint, stably-infinite theories* T_1 and T_2 for which we have theory solvers which run in $O(\mathrm{T}_1)$ and $O(\mathrm{T}_2)$:
- The combined theory $\mathrm{T}_1 \bigcup \mathrm{T}_2$ is stably infinite
- For a conjunction of *literals* (note: not clauses) in the combined theory, satisfiability can be decided in exponential time
  - by non-deterministic Nelson-Oppen
  - exponential comes from number of equivalence relations to “guess”
- If both theories are convex, the same question can be decided in polynomial time (deterministic Nelson-Oppen)
- Note that these complexities do not account for the overall DPLL search, involved in answering SMT for general CNF formulas

SMT Solving Algorithms - Summary

- We have seen how to extend DPLL/CDCL with a single theory solver
- We have also seen how to combine multiple theories (Nelson-Oppen)
  - some restrictions on the theories, but many theories satisfy these
  - the combination of two such theories satisfies the restrictions
- For the special case of convex theories, deterministic Nelson-Oppen is advantageous: allows for incremental cooperation between solvers
  - Leads to a lot of flexibility in extending approach (e.g. learning/deduction)
- So far, we have only considered quantifier-free formulas
- In the next section: how to incorporate first-order quantification

SMT Solving Algorithms – Some References

- *Handbook of Satisfiability.* Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh (2009)
  - relevant chapter available from: `http://ebooks.iospress.nl/publication/5011`
- Nelson-Oppen: *Simplification by Cooperating Decision Procedures.* Greg Nelson, Derek C. Oppen (1979).
- Theory Integration: *DPLL(T): Fast Decision Procedures.* Harald Ganzinger, George Hagen, Robert Nieuwenhuis, Albert Oliveras, Cesare Tinelli (2004)
- Other teaching material:
  - *SMT Solvers: Theory and Implementation.* Leonardo de Moura
  - *SMT Solvers: Theory and Practice.* Clark Barrett
- SMT-solving Competition: *SMT-COMP 2016.* Sylvain Conchon, David Déharbe, Matthias Heizmann, Tjark Weber