

Decision Procedures

Jochen Hoenicke



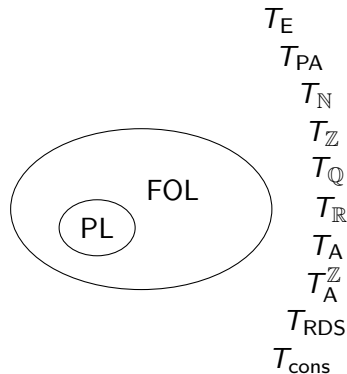
Software Engineering
Albert-Ludwigs-University Freiburg

Summer 2012

Conclusion

Lecture	Topics
1–3	Propositional Logic
4–5	First-Order Logic
6–8	First-Order Theories
9–10	Quantifier Elimination for $T_{\mathbb{Z}}$ and $T_{\mathbb{Q}}$
11–12	Dutertre–de Moura Algorithm ($T_{\mathbb{Q}}$)
13–14	Congruence Closure Algorithm ($T_E, T_{\text{cons}}, T_A$)
15–16	DP for Array Property Fragment
17–18	Nelson-Oppen
19–20	DPLL(T) with Learning
21–22	Program Correctness
23	Interpolation

PL	Propositional Logic	Lec. 1–3
FOL	First-Order Logic	Lec. 4–5
T_x	Theories	Lec. 6–8



Theory	Full	Array Prop.	Quant. free	Conj. quant. free
T_E	✗	-	✓ (19–20)	✓ (13)
T_{PA}	✗	-	✗	✗
T_Z	✓ (10)	-	✓	✓
T_Q	✓ (9)	-	✓ (19–20)	✓ (11–12)
T_R	✓ (-)	-	✓	✓
T_A	✗	✓ (15–16)	✓	✓ (15)
T_A^Z	✗	✓ (15–16)	✓	✓
T_{RDS}	✗	-	✓	✓
T_{cons}	✗	-	✓	✓ (14)
$T_1 \cup T_2$	-	-	✓ (-)	✓ (17–18)

- What is an **atom**, a **literal**, a **formula**.
- What is an **interpretation**?
- What does $I \models F$ mean, how do we compute it.
- What is **satisfiability**, **validity**.
- What is the duality between satisfiable and valid?
- What is the **semantic argument**?
- Write down the **proof rules**.
- How can we prove $P \wedge Q \rightarrow P \vee \neg Q$?
- What is \Leftrightarrow (**equivalent**) and \Rightarrow (**implies**).
- What **Normal Forms** do you know (**NNF**, **DNF**, **CNF**)?
- How to convert formulae into normal form.

- What is a **Decision Procedure**?
- What is **equisatisfiability**; why is it useful?
- How to convert to CNF with **polynomial** time complexity?
- What is a **clause**?
- What does DPLL stand for?
- What is **Boolean Constraint Propagation** (BCP) (aka. Unit Propagation).
- What is **Pure Literal Propagation** (PL).
- Why is the DPLL algorithm correct?
- What is the worst case time complexity of DPLL?

- What is a **variable**, a **constant**, a **function (symbol)**, a **predicate (symbol)**, a **term**, an **atom**, a **literal**, a **formula**?
- How do first-order logic and predicate logic relate?
- What is an **interpretation** in FOL?
- Why is D_I non-empty?
- What does α_I assign?
- What is an **x-variant** of an interpretation?
- How do we compute whether $I \models F$?
- What is **satisfiability**, **validity**?
- What are the additional rules in the **Semantic Argument** (version of lecture 4)?
- **Soundness** and **Completeness** of semantic argument.
- What is a **Hintikka set**?
- Normal forms. What is **PNF** (**prenex normal form**)?
- Is **validity** for FOL **decidable**?

- What is a **theory**?
- What is a **signature** Σ ?
- What do **T -valid** and **emph T -satisfiable** mean?
- What is **T -equivalent**?
- What is a **decision procedure** for a theory?
- What is a **fragment** of a theory?
- What are the most common fragments (**quantifier-free, conjunctive**)?
- What **theories** do you know?
- What are their **axioms**?
- What **fragments** of these theories are **decidable**?
- Bonus Question: Is there any closed formulae in T_{PA} that is satisfiable but not valid? What about $T_{\mathbb{Z}}$, $T_{\mathbb{Q}}$?

- What is **Quantifier Elimination**?
- Does $T_{\mathbb{Z}}$ **admit** quantifier elimination? What does it mean?
- Why is it enough to eliminate **one** existential quantifiers over a **quantifier-free formula**?
- How can we eliminate more than one quantifier?
- What is $\widehat{T}_{\mathbb{Z}}$?
- What is **Cooper's method**?
- What is **Ferrante and Rackoff's method** ($T_{\mathbb{Q}}$)?
- What is the **Array Property Fragment**?
- What do all quantifier elimination methods of the lecture **have in common**?
- What is the **complexity** of quantifier elimination?
- Why is quantifier elimination a **decision procedure**?

- Which theory does the Algorithm of [Dutertre and de Moura](#) decide?
- How does the algorithm work?
- How can we convert an [arbitrary formula](#) to the required format for the algorithm?
- What is the tableaux?
- What is a pivot step?
- Does the algorithm [terminate](#)?
- What is the [complexity](#)?

- What is the **congruence closure algorithm**?
- How does it work for T_E ?
- What are the data structures; what are the operations?
- What **complexity** does the algorithm have?
- What are the extensions for T_{cons} ?
- What is the **complexity**?
- How did we prove **correctness** of the decision procedure?

- How does the DP for **quantifier-free fragment** of T_A work?
- What is the **complexity**?
- What is $T_A^=$?

- What is the **Array Property Fragment** of $T_A/T_A^=$?
- Why are there so many restrictions?
- What are the transformation steps?
- How are quantifier eliminated?
- What is λ and why is it necessary?
- Why is the decision procedure correct?
- What is the **Array Property Fragment** of $T_A^{\mathbb{Z}}$?
- What are differences to T_A ?
- Why do we not need λ for $T_A^{\mathbb{Z}}$?
- Why is the decision procedure correct?
- How can we check this fragment?

- What is the **Nelson-Oppen** procedure?
- For what theories does it work? For which fragment of the theory?
- What is a **stably infinite** theory?
- Why is it important that theories are stably infinite?
- What are the two phases of Nelson-Oppen?
- What is the difference between the **non-deterministic** and **deterministic** variant of Nelson-Oppen?
- What is a **convex** theory?
- What is the **emphcomplexity** of the deterministic version for convex/non-convex theories?

- How can we extend the DPLL algorithm to decide T -satisfiability.
- What is a **minimal unsatisfiable core**?
- How can we compute it efficiently?
- What is the relation between min. unsat. core and conflict clause?
- Why is the algorithm **correct**, why does it **terminate**?
- How can we extend it to **more than one** theory?
- What is the relation to Nelson-Oppen?

- What is a **specification**?
- What types of specification are in a typical program?
(**Precondition**, **postcondition**, **loop invariants**, **assertions**)
- When is a procedure correct (**partial/total correctness**)?
- What is a **basic path**? Why is it useful?
- How do we prove correctness of a basic path?
- What is a **verification condition**?
- What is the **weakest precondition**?
- How do we compute weakest precondition?
- What is a **P -invariant** annotation, what is a **P -inductive** annotation?
- Why are we interested in P -inductive annotations?
- What is a **ranking function**? Why do we need it?
- What is a **well-founded relation**?
- How do we prove **total correctness**?

- You should learn **definitions** (formally).
This includes the **rules** (semantic argument, DPLL with learning).
- You should **understand** them (informally).
- You should know **important theorems**.
- Knowing the proofs is a plus. Don't lose yourself in the details!
- You should be able to **apply** the decision procedures.
Do the exercises! Invent some new exercises and solve them!
- You should know some examples/counter-examples,
e.g., why is λ necessary?
- When you feel well prepared, check if you can answer the questions in this slide set.
- When learning, do not leave out a whole topic completely!
- Learn in a group. Ask questions of each other and answer them as if you were in the exam.

- There will be only **oral exams** for this lecture.
- You should have officially registered at the Prüfungsamt.
- The exams will be in September.