Seminar

Decision Procedures and Applications

Instructor: Viorica Sofronie-Stokkermans

Universität Koblenz-Landau

`http://userpages.uni-koblenz.de/~sofronie/sem-decproc-ss-2019/`

Motivation

Long-term goal of research in computer science

- use computers as 'intelligent assistants' in

  e.g. mathematics, engineering (and other fields)

Main problem

- complex description of problems to be solved

  $\mapsto$ complex systems, complex encoding

Examples of application domains

MATHEMATICS

Tasks
- construct proofs
- check proofs

Theories
- numbers
- polynomials
- functions over numeric domains
- algebras

VERIFICATION

Tasks
- **programs**
  - correctness
  - termination
- **reactive/hybrid systems**
  - safety / lifeness

Theories
- numbers
- data types
- functions over numeric domains

DATA BASES

Tasks
- test consistency
- answer queries
- limit search

Theories
- First−order logic
- Datalog
- ...

Complex theories
- numbers
- functions

complex systems (MAS, reactive systems w. embedded software, databases)

Examples of application domains

MATHEMATICS

Tasks
- construct proofs
- check proofs

Theories
- numbers
- p
- f
  n
- algebras

VERIFICATION

Tasks
- **programs**
  - correctness
  - termination
- **reactive/hybrid systems**
  - safety / lifeness

DATA BASES

Tasks
- test consistency
- answer queries
- limit search

Theories
- First−order logic

Method: - encode problems as logical formulae

- test entailment / satisfiability / validity

complex systems (MAS, reactive systems w. embedded software, databases)

Problems and goals

– 1^{st} order logic is undecidable: cannot build an 'all-purpose' program

\+ often fragments of theories occurring in applications are decidable

– theories do not occur alone: need to consider combinations of theories

\+ often provers for the component theories can be combined efficiently

Important:

Identify theories (and extensions/combinations thereof) which are decidable (with low complexity) and relevant in applications

Goal of the seminar

- Identify decidable/tractable fragments of 1^{st}-order logic

- Discuss methods for proving decidability of logical theories

- Identify application domains where decision procedures are used.

Overview

- **Reasoning in first-order logic**
- **Reasoning about standard datatypes**
- **Reasoning in theory extensions**
- **Reasoning in combinations of theories**

Important: identify decidable/tractable fragments

... important for practical applications

(verification, databases, ...)

Reasoning in first-order logic

In 1931, Gödel published his incompleteness theorems in

"Über formal unentscheidbare Sätze der
Principia Mathematica und verwandter Systeme"

(in English "On Formally Undecidable Propositions of Principia Mathematica and Related Systems").

He proved for any computable axiomatic system that is powerful enough to describe the arithmetic of the natural numbers (e.g. the Peano axioms or Zermelo-Fraenkel set theory with the axiom of choice), that:

- If the system is consistent, it cannot be complete.
- The consistency of the axioms cannot be proven within the system.

Decidability/Undecidability

These theorems ended a half-century of attempts, beginning with the work of Frege and culminating in Principia Mathematica and Hilbert's formalism, to find a set of axioms sufficient for all mathematics.

The incompleteness theorems also imply that not all mathematical questions are computable.

Consequences of Gödel's Famous Theorems

1. For most signatures Σ, validity is undecidable for Σ-formulas. (One can easily encode Turing machines in most signatures.)
2. For each signature Σ, the set of valid Σ-formulas is recursively enumerable.
   (We will prove this by giving complete deduction systems.)
3. For $\Sigma = \Sigma_{PA}$ and $\mathbb{N}_* = (\mathbb{N}, 0, s, +, *)$, the theory $Th(\mathbb{N}_*)$ is not recursively enumerable.

These undecidability results motivate the study of subclasses of formulas (fragments) of first-order logic

Some Decidable Fragments/Problems

Validity/Satisfiability/Entailment: Some decidable fragments:

- Variable-free formulas without equality:
  satisfiability is NP-complete. (why?)
- Variable-free Horn clauses (clauses with at most one positive atom):
  entailment is decidable in linear time.
- Monadic class: no function symbols, all predicates unary;
  validity is NEXPTIME-complete.
- Other decidable fragments of FOL (with variables):

  Ackermann class
  Bernays Schönfinkel class
  Guarded fragment

  Methods for proving decidability: "small model" theorems
  for some classes also resolution

Logical theories

Syntactic view

first-order theory: given by a set $\mathcal{F}$ of (closed) first-order Σ-formulae.

the models of $\mathcal{F}$: $\quad \mathsf{Mod}(\mathcal{F}) = \{\mathcal{A} \in \Sigma\text{-alg} \mid \mathcal{A} \models G, \text{ for all } G \text{ in } \mathcal{F}\}$

Semantic view

given a class $\mathcal{M}$ of Σ-algebras

the first-order theory of $\mathcal{M}$: $\mathsf{Th}(\mathcal{M}) = \{G \in F_\Sigma(X) \text{ closed} \mid \mathcal{M} \models G\}$

Decidable theories

Let $\Sigma = (\Omega, \Pi)$ be a signature.

$\mathcal{M}$: class of Σ-algebras. $\mathcal{T} = \mathrm{Th}(\mathcal{M})$ is decidable

iff

there is an algorithm which, for every closed first-order formula ϕ, can decide (after a finite number of steps) whether ϕ is in $\mathcal{T}$ or not.

$\mathcal{F}$: class of (closed) first-order formulae.

The theory $\mathcal{T} = \mathrm{Th}(\mathrm{Mod}(\mathcal{F}))$ is decidable

iff

there is an algorithm which, for every closed first-order formula ϕ, can decide (in finite time) whether $\mathcal{F} \models \phi$ or not.

Decidable theories

- Presburger arithmetic decidable in 3EXPTIME [Presburger'29]
  Signature: $(\{0, 1, +\}, \{\approx, \leq\})$ (no $*$)

  Axioms { (zero), (successor), (induction), (plus zero), (plus successor) }

- Th($\mathbb{Z}_+$) $\mathbb{Z}_+ = (\mathbb{Z}, 0, s, +, \leq)$ the standard interpretation of integers.
- The theory of real numbers (with addition and multiplication) is decidable in 2EXPTIME [Tarski'30]

Undecidable theories:

- Th($(\mathbb{Z}, \{0, 1, +, *\}, \{\leq\})$)
- Th(Σ-alg)

Decidability results for certain fragments

$\mathcal{T}$: first-order theory in signature Σ; $\mathcal{L}$ class of (closed) Σ-formulae

Given ϕ in $\mathcal{L}$, is it the case that $\mathcal{T} \models \phi$?

Common restrictions on $\mathcal{L}$

	$\text{Pred} = \emptyset$	$\{\phi \in \mathcal{L} \mid \mathcal{T} \models \phi\}$
$\mathcal{L}=\{\forall x A(x) \mid A \text{ atomic}\}$	word problem	
$\mathcal{L}=\{\forall x (A_1 \wedge \ldots \wedge A_n \rightarrow B) \mid A_i, B \text{ atomic}\}$	uniform word problem	$\text{Th}_{\forall\text{Horn}}$
$\mathcal{L}=\{\forall x C(x) \mid C(x) \text{ clause}\}$	clausal validity problem	$\text{Th}_{\forall,\text{cl}}$
$\mathcal{L}=\{\forall x \phi(x) \mid \phi(x) \text{ unquantified}\}$	universal validity problem	$\text{Th}_{\forall}$
$\mathcal{L}=\{\exists x A_1 \wedge \ldots \wedge A_n \mid A_i \text{ atomic}\}$	unification problem	$\text{Th}_{\exists}$
$\mathcal{L}=\{\forall x \exists x A_1 \wedge \ldots \wedge A_n \mid A_i \text{ atomic}\}$	unification with constants	$\text{Th}_{\forall\exists}$

Application domains

MATHEMATICS

Tasks
- construct proofs
- check proofs

Theories
- numbers
- polynomials
- functions over numeric domains
- algebras

VERIFICATION

Tasks
- **programs**
  - correctness
  - termination
- **reactive/hybrid systems**
  - safety / lifeness

Theories
- numbers
- data types
- functions over numeric domains

DATA BASES

Tasks
- test consistency
- answer queries
- limit search

Theories
- First−order logic
- Datalog
- ...

Complex theories
- numbers
- functions

Examples of application domains

MATHEMATICS

Tasks

- construct proofs
- check proofs

Theories

- numbers
- polynomials
- functions over numeric domains
- algebras

Example: Lipschitz functions

$$\mathbb{R} \cup (L^{f}_{c,\lambda_1}) \cup (L^{g}_{c,\lambda_2}) \models (L^{f+g}_{c,(\lambda_1+\lambda_2)})$$

(L^{f}_{c,λ_1}) $\forall x \; |f(x) - f(c)| \leq \lambda_1 \cdot |x - c|$

(L^{g}_{c,λ_2}) $\forall x \; |g(x) - g(c)| \leq \lambda_2 \cdot |x - c|$

$(L^{f+g}_{c,(\lambda_1+\lambda_2)})$ $\forall x \; |f(x)+g(x)-f(c)-g(c)| \leq (\lambda_1+\lambda_2) \cdot |x-c|$

Similar:

- free functions; (piecewise) monotone functions
- functions defined according to a partition of their domain of definition, ...

Examples of application domains

MATHEMATICS

Tasks
- construct proofs
- check proofs

Theories
- numbers
- polynomials
- functions over numeric domains
- algebras

VERIFICATION

Tasks
- **reactive and hybrid systems**
  - safety / lifeness
- **programs**
  - correctness
  - termination

Infinite state systems (software, real time, hybrid)

- simulation/testing cannot guarantee absence of errors

$\mapsto$ need symbolic methods

Solution: - Build 'formal model' of the system;

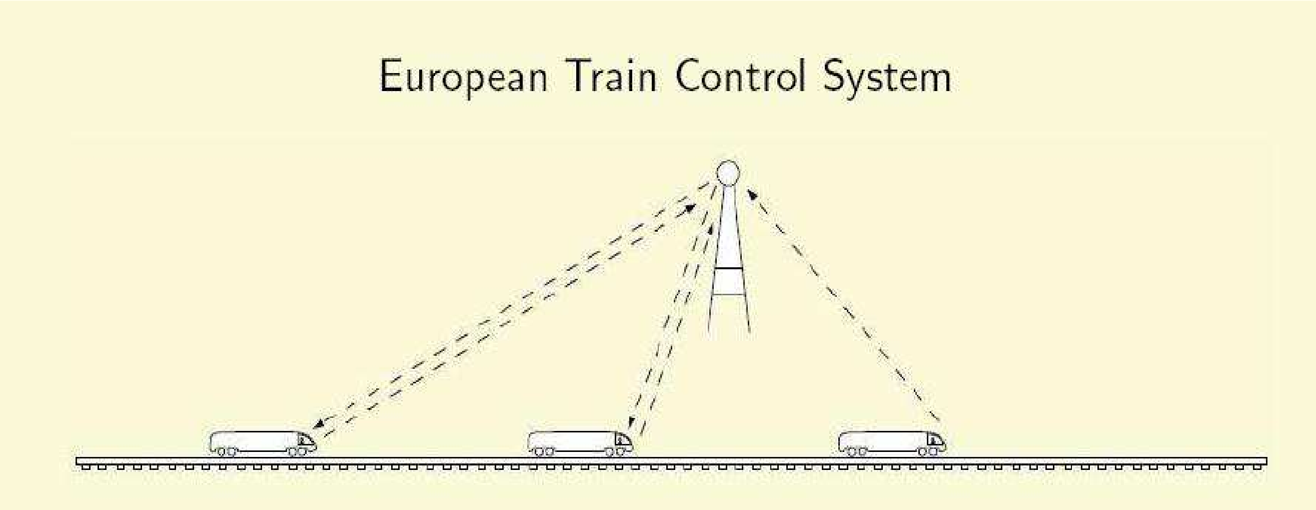
- Prove that properties are 'consequences of the model'

Decision Procedures for Verification

- Verification of train controllers
- Program verification

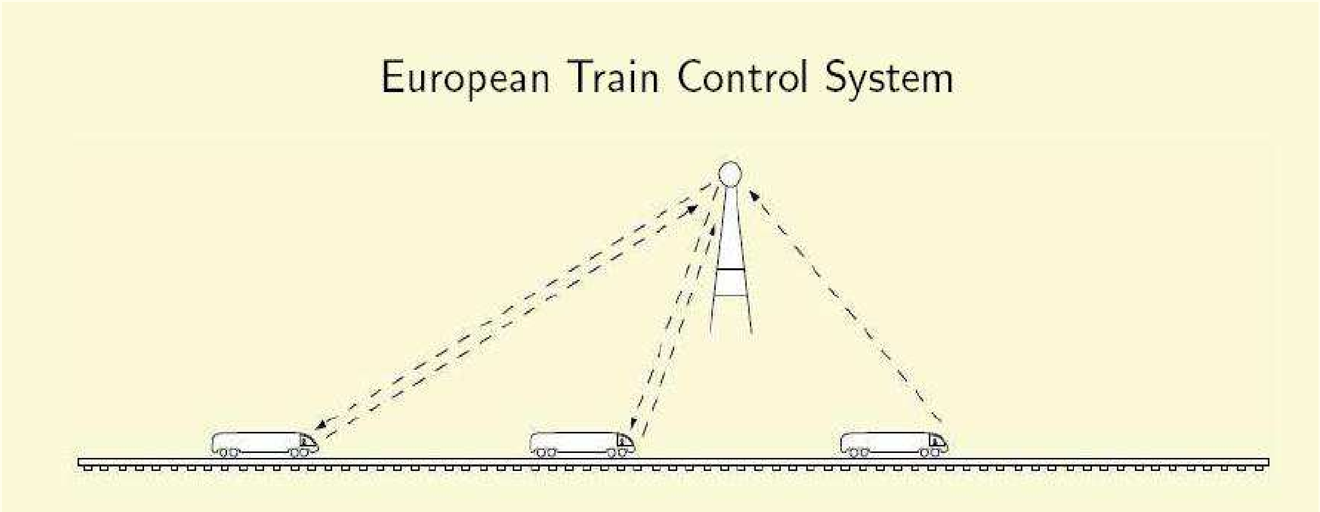
Methods for reasoning in theories of datatypes: extremely important.

Example 1: Train control system



Number of trains:	$n \geq 0$	$\mathbb{Z}$
Minimum and maximum speed of trains:	$0 \leq \text{min} < \text{max}$	$\mathbb{R}$
Minimum secure distance:	$l_{\text{alarm}} > 0$	$\mathbb{R}$
Time between updates:	$\Delta t > 0$	$\mathbb{R}$
Train positions before and after update:	$pos(i), pos'(i)$	$: \mathbb{Z} \to \mathbb{R}$

Example 1: Train control system



Update(pos, pos′) :

- $\forall i\ (i = 0 \rightarrow pos(i) + \Delta t{*}\text{min} \leq pos'(i) \leq pos(i) + \Delta t{*}\text{max})$
- $\forall i\ (0 < i < n\ \wedge\ pos(i-1) > 0\ \wedge\ pos(i-1) - pos(i) \geq l_{\text{alarm}}$
  $\rightarrow pos(i) + \Delta t * \text{min} \leq pos'(i) \leq pos(i) + \Delta t{*}\text{max})$

...

Example 1: Train control system

Safety property: No collisions

$\mathsf{Safe}(\mathsf{pos}): \quad \forall i, j (i < j \rightarrow \mathsf{pos}(i) > \mathsf{pos}(j))$

Inductive invariant:

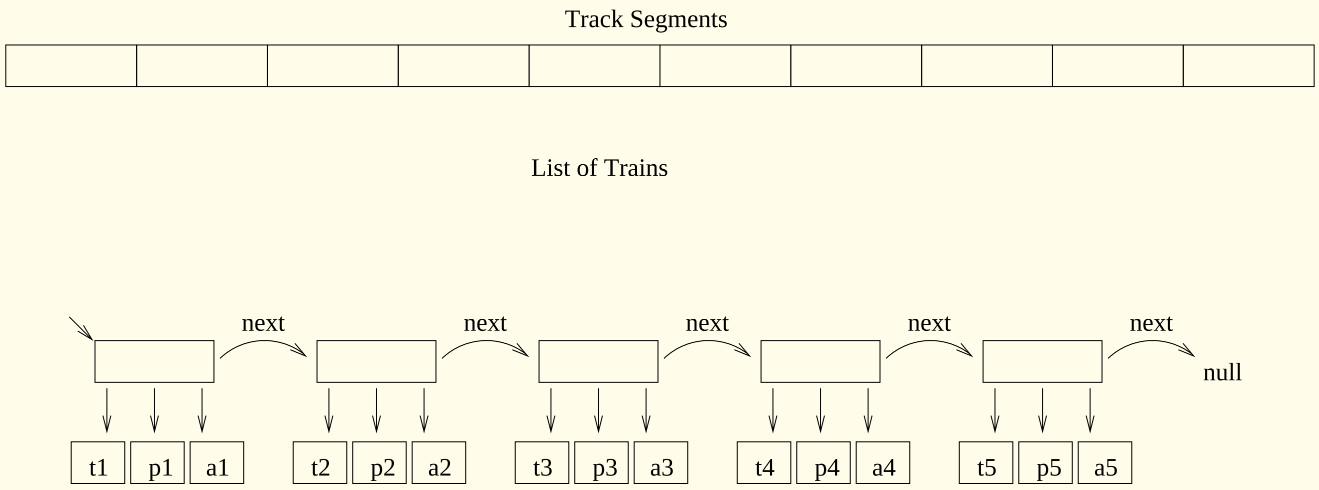
$$\mathsf{Safe}(\mathsf{pos}) \wedge \mathsf{Update}(\mathsf{pos}, \mathsf{pos}') \wedge \neg \mathsf{Safe}(\mathsf{pos}') \models_{\mathcal{T}_S} \perp$$

where $\mathcal{T}_S$ is the extension of the (disjoint) combination $\mathbb{R} \cup \mathbb{Z}$ with two functions, $\mathsf{pos}, \mathsf{pos}' : \mathbb{Z} \rightarrow \mathbb{R}$

Problem: Satisfiability test for quantified formulae in complex theory

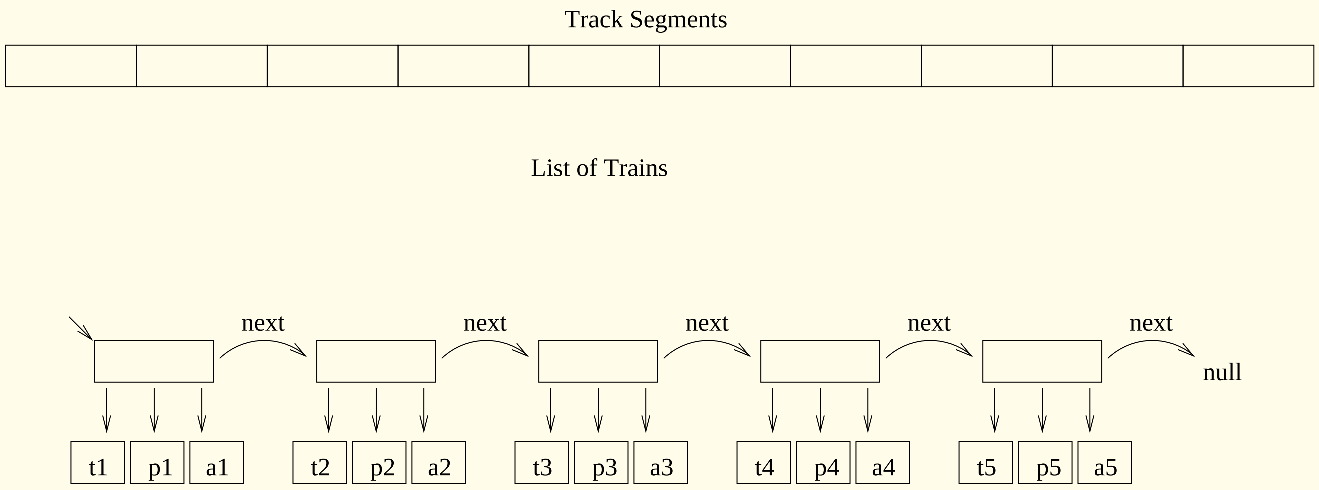
Example 1: Train control system

Various track segments



Example: Train control system

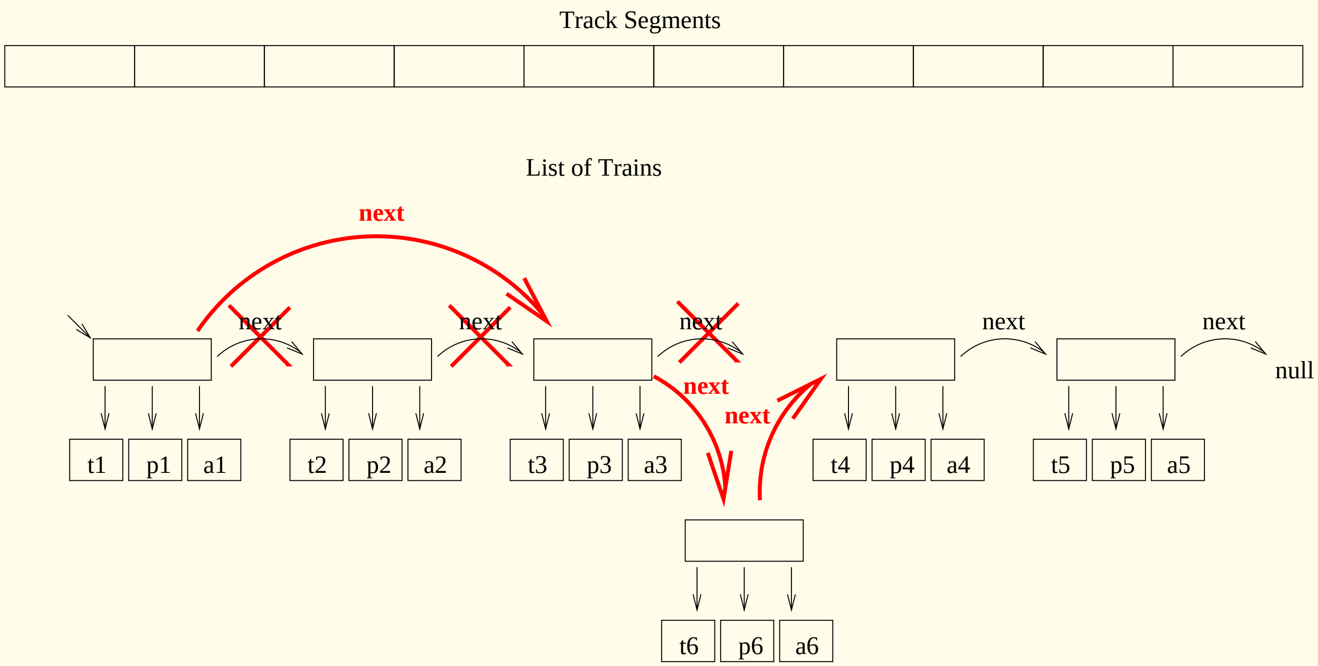
Various track segments



Data structure: Lists

Example 1: Train control system

Various track segments

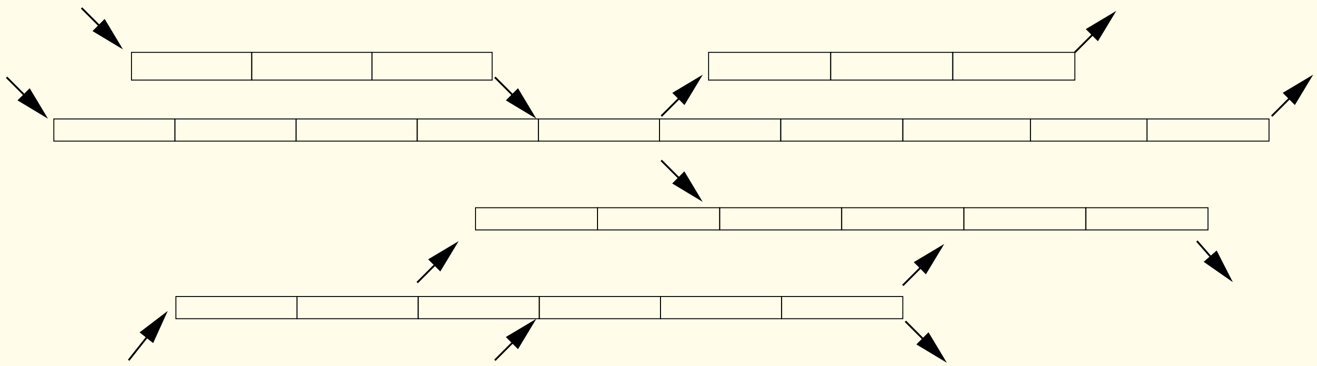


Data structure: Lists

Operations: Insert/Delete

Example: Train control system

Complex track system



Example 2

$-1 \leq i < |a| \wedge$
partitioned$(a, 0, i, i+1, |a|-1) \wedge$
sorted$(a, i, |a|-1)$

$-1 \leq i < |a| \wedge 0 \leq j \leq i \wedge$
partitioned$(a, 0, i, i+1, |a|-1) \wedge$
sorted$(a, i, |a|-1)$
partinioned$(a, 0, j-1, j, j)$

Example: Does BubbleSort return a sorted array?

```
int [] BubbleSort(int[] a) {
  int i, j, t;
  for (i := |a| − 1; i > 0; i := i − 1) {
    for (j := 0; j < i; j := j + 1) {
      if (a[j] > a[j + 1]){t := a[j];
                           a[j] := a[j + 1];
                           a[j + 1] := t};
}} return a}
```

Generate verification conditions and prove that they are valid

Predicates:

- sorted(a, l, u): $\forall i, j(l \leq i \leq j \leq u \rightarrow a[i] \leq a[j])$
- partitioned(a, l_1, u_1, l_2, u_2): $\forall i, j(l_1 \leq i \leq u_1 \leq l_2 \leq j \leq u_2 \rightarrow a[i] \leq a[j])$

Example 2

$-1 \leq i < |a| \wedge$ $C_1(a)$

partitioned$(a, 0, i, i+1, |a| - 1) \wedge$

sorted$(a, i, |a| - 1)$

$-1 \leq i < |a| \wedge 0 \leq j \leq i \wedge$ $C_2(a)$

partitioned$(a, 0, i, i+1, |a| - 1) \wedge$

sorted$(a, i, |a| - 1)$

partinioned$(a, 0, j-1, j, j)$

Example: Does BubbleSort return a sorted array?

```
int [] BubbleSort(int[] a) {
  int i, j, t;
  for (i := |a| - 1; i > 0; i := i - 1) {
    for (j := 0; j < i; j := j + 1) {
      if (a[j] > a[j + 1]){t := a[j];
                           a[j] := a[j + 1];
                           a[j + 1] := t};
}} return a}
```

Generate verification conditions and prove that they are valid

$C_2(a) \wedge \text{Update}(a, a') \rightarrow C_2(a')$

Other examples

- Knowledge representation:

  Relational databases, terminological databases.

  Non-classical logics

- ....

Reasoning about standard datatypes

- **Numbers** - natural numbers, integers, reals, rationals
- **Data structures** - theories of lists
  - theory of acyclic lists
  - theory of arrays
  - theories of sets, multisets
- **Fragments of FOL**

Reasoning in theory extensions

- **Numbers** - integers, reals, rationals

- **Data structures**
  - theories of lists of integers, reals, ...
  - theory of acyclic lists of integers, reals, ...
  - theory of arrays of integers, reals, ...
  - theories of sets of integers, reals, ...

  + functions (free, rec. def.) e.g : length, card

- **Fragments of FOL** + a certain amount of arithmetic/other datatypes

Extensions & combinations of theories

Train controller example: Reason about - arrays of real numbers
- lists (+ next) with scalar fields

Program verification: Reason about lists (arrays, sets) over some data

Mathematics: Reason about properties of real functions

Databases, knowledge representation: Reason in fragments of FOL + arithmetic (+ other datatypes)

Ideally: Use a prover for the base theory as a black-box

Extensions & combinations of theories

Program verification: Reason about reals, lists, and (free) functions

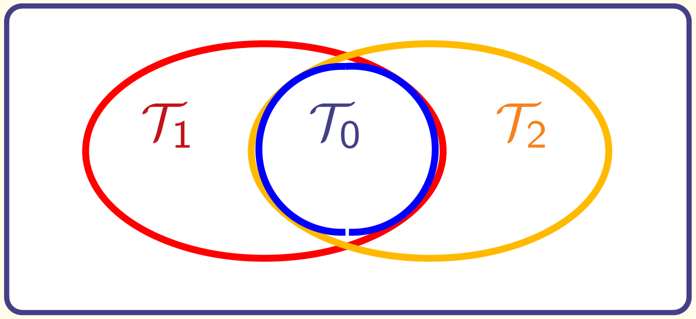
Reason about lists and arrays over some data

Logic: Reason in combinations of modal logics

Reason in combinations of (logical) databases

Ideally: Use provers for the components as black-boxes

Combinations of Theories



Which information needs to be exchanged between provers for the component theories to guarantee completeness?

Link with **interpolation**

$A \wedge B \models \perp \quad \Rightarrow \quad \exists I$ containing only symbols common to A, B

with $A \models_{\mathcal{T}_1} I$ and $I \wedge B \models_{\mathcal{T}_2} \perp$

Applications:

- Verification
- (Distributed) databases
- Logic

Structure of the seminar

Time and place: Tuesday: 12:00-14:00, Room C 209

change necessary?

Structure

- General comments: talks and presentations
- Background: first-order logic, theorem proving, combinations of theories (generalities, motivation)
- Presentations: $\sim$ 45 min (30 min talk + up to 15 min discussions), slides, written abstract

Topics

- List of suggested topics + literature (+ Additional possible topics)

Further possibilities

- continue with a BSc or Master thesis (topic of the talk, or other topic).

Topics

Decidable fragments of first-order logic

some examples:

Monadic first-order logic, Bernays-Schönfinkel class, Ackermann class, Guarded fragment, ...

Small model property/Finite model property

Resolution-based decision procedures

Topics

Decision procedures for data types

Simple data types

1. Reasoning about uninterpreted function symbols; congruence closure

- G. Nelson and D.C. Oppen. Fast decision procedures based on congruence closure. Journal of the ACM, 27(2):356-364, 1980.
- L. Bachmair and A. Tiwari and L. Vigneron. Abstract Congruence Closure. J. of Automated Reasoning 31(2), 2003, pp. 129–168, Kluwer Academic Publishers.

2. Superposition decision procedures for data types (lists, arrays, ...)

- Alessandro Armando, Silvio Ranise, Michaël Rusinowitch. A rewriting approach to satisfiability procedures. Inf. Comput. 183(2): 140-164 (2003)

Topics

Decision procedures for data types

Numerical domains

3. Linear integer (Presburger) arithmetic: The automata-theoretic method

- J.E. Hopcroft and J.D. Ullmann. Introduction to Automata Theory, Languages, and Computation, chapter 2: Finite automata and regular expressions, pages 13-54. Addison-Wesley, 1979.
- H. Comon and C. Kirchner. Presburger arithmetic and classical word automata. In H. Comon, C. MarchÃÂ© and R. Treinen, editors, Constraints in Computational Logic: Theory and Applications, volume 2002 of LNCS, chapter 2: Constraint solving on terms, section 8, pages 79-83. Springer-Verlag, 1999.

Topics

Decision procedures for data types

Numerical domains

4. Difference Logic and UTVPI Constraints

- An Efficient Decision Procedure for UTVPI Constraints Shuvendu K. Lahiri Madanlal Musuvathi Proc. FroCoS 2005, LNCS 3717, pp.168-183, Springer 2005.

  (see also https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/tr-2005-67.pdf)

- several other papers can be used.

Topics

Reasoning in complex theories

5. Combinations of theories over disjoint signatures

- G. Nelson and D.C. Oppen. Simplification by cooperating decision procedures. ACM Transactions on Programming Languages and Systems, 1(2):243-257, 1979.
- Derek C. Oppen. Complexity, Convexity and Combinations of Theories. Theor. Comput. Sci. 12: 291-302, 1980.
- C. Tinelli and M. Harandi. A new correctness proof of the Nelson-Oppen combination procedure. Proceedings FroCos'96.
- Recent results refine the combination method.

6. Combinations of theories over non-disjoint signatures

- S. Ghilardi. Model Theoretic Methods in Combined Constraint Satisfiability. Journal of Automated Reasoning, vol. 33, no. 3-4, pp.221-249 (2005).

Topics

Reasoning in theory extensions

7. Local theory extensions

- S. Burris Polynomial time uniform word problems. Math. Logic Quarterly 41 (1995), 173 - 182.
- H. Ganzinger. Relating Semantic and Proof-Theoretic Concepts for Polynomial Time Decidability. Proc. 16th IEEE Symposium on Logic in Computer Science, pages 81–92, IEEE Computer Society Press, 2001.
- Viorica Sofronie-Stokkermans. Hierarchic reasoning in local theory extensions. Proceedings of the 20th International Conference on Automated Deduction (CADE 2005), (R. Nieuwenhuis, ed.), LNAI 3632, pages 219-234, Springer Verlag, 2005.
- Several recent results on applications

Topics

Reasoning about complex data types

8. Instantiation-based decision procedures for theories of arrays.

- Aaron Bradley, Zohar Manna, Henny Sipma: What's decidable about arrays? Proceedings VMCAI 2006.
- Silvio Ghilardi, Enrica Nicolini, Silvio Ranise, Daniele Zucchelli:
  Decision procedures for extensions of the theory of arrays. Ann. Math. Artif. Intell. 50(3-4): 231-254 (2007)
- Silvio Ghilardi, Enrica Nicolini, Silvio Ranise, Daniele Zucchelli:
  Deciding Extensions of the Theory of Arrays by Integrating Decision Procedures and Instantiation Strategies.
  JELIA 2006: 177-189
- Francesco Alberti, Silvio Ghilardi, Natasha Sharygina:
  Decision Procedures for Flat Array Properties. J. Autom. Reasoning 54(4): 327-352 (2015)

Topics

Reasoning about complex data types ctd.

9. Decision procedures for recursive data structures with integer constraints

- Ting Zhang, Henny Sipma, Zohar Manna, Decision procedures for recursive data structures with integer constraints. Proceedings IJCAR 2004;

- Ting Zhang, Henny Sipma, Zohar Manna, Decision procedures for recursive data structures with integer constraints. Information and Computation 2006.

Topics

Reasoning about complex data types ctd.

10. Decision procedures for sets with cardinalities

- Hans Jürgen Ohlbach: Set Description Languages and Reasoning about Numerical Features of Sets. Description Logics 1999
- Hans Jürgen Ohlbach, Jana Koehler: Modal Logics, Description Logics and Arithmetic Reasoning. Artif. Intell. 109(1-2): 1-31 (1999)
- Viktor Kuncak, Huu Hai Nguyen, Martin C. Rinard. Deciding Boolean Algebra with Presburger Arithmetic. J. Autom. Reasoning 36(3): 213-239 (2006)

Topics

SAT checking modulo a theory

11. *DPLL*($\mathcal{T}$)

- Harald Ganzinger, George Hagen, Robert Nieuwenhuis, Albert Oliveras, Cesare Tinelli. DPLL(T): Fast Decision Procedures . CAV 2004: 175-188
- Robert Nieuwenhuis, Albert Oliveras. DPLL(T) with Exhaustive Theory Propagation and Its Application to Difference Logic. CAV 2005: 321-334

Extensions:

- non-chronological backtracking

- conflict-driven clause learning (CDCL)

- Joao Marques-Silva, Ines Lynce and Sharad Malik: Conflict-Driven Clause Learning SAT Solvers. Handbook of Satisfiability, 2009.

Topics

Interpolation

12. Interpolation in extensions of linear arithmetic with free functions and/or in extensions and combinations of theories

- Ken McMillan. An interpolating theorem prover. Theoretical Computer Science, 2005.
- A. Rybalchenko, V.Sofronie-Stokkermans. Constraint solving for interpolation, VMCAI 2006.

- G. Yorsh, M. Musuvathi. A Combination Method for Generating Interpolants. Proc. CADE 2005.
- V.Sofronie-Stokkermans. Interpolation in local theory extensions. Proc. IJCAR 2006.
- Roberto Bruttomesso, Silvio Ghilardi, Silvio Ranise: From Strong Amalgamability to Modularity of Quantifier-Free Interpolation. IJCAR 2012: 118-133

Topics

Applications to verification

13. Invariant checking; Bounded model checking

- Carsten Ihlemann, Swen Jacobs, Viorica Sofronie-Stokkermans: On local reasoning in verification, Proceedings TACAS 2008.
- Leonardo de Moura, Harald Ruess and Maria Sorea. Lazy Theorem Proving for Bounded Model Checking over Infinite Domains Proceedings of CADE 18, LNCS 2392, pages 438–455, 2002.
- several other papers on the topic

14. Verification by abstraction/refinement

- Ken McMillan. Application of Craig interpolation in model checking, Proc. TACAS 2005.

(many additional papers - see also the website)

Topics

Applications to knowledge representation

15.– 20. Various possible topics on modal logics, description logics; combinations of modal logics; distributed databases.

PTIME fragment of Description logics

- Franz Baader, Sebastian Brandt, Carsten Lutz: Pushing the EL Envelope. IJCAI 2005: 364-369
- F. Baader, C. Lutz, B. Suntisrivaraporn. Is Tractable Reasoning in Extensions of the Description Logic EL Useful in Practice?; Journal of Logic, Language and Information (M4M special issue); 2007.

Various papers on unification in EL and applications to databases/ontologies

....