Fundamentals of Programming Languages IX

Conclusion

Guoqiang Li

School of Software, Shanghai Jiao Tong University

Report

The deadline is firmly on Jan. 10, 2018!

Report

The deadline is firmly on Jan. 10, 2018!

Both paper version and electronic version are needed (3203 Software Building & li.g@outlook.com)!

Scoring Policy

- 10% Attendance.
- 20% Homework.
  - Four assignments.
  - Each one is 5pts.
  - Work out individually.
  - Each assignment will be evaluated by A, B, C, D, F (Excellent(5), Good(5), Fair(4), Delay(3), Fail(0))
- 70% Final report.

Scoring Policy

- 10% Attendance.
- 20% Homework.
  - Four assignments.
  - Each one is 5pts.
  - Work out individually.
  - Each assignment will be evaluated by A, B, C, D, F (Excellent(5), Good(5), Fair(4), Delay(3), Fail(0))
- 70% Final report.

Homework is MUCH MORE IMPORTANT than report!

What We Have Learnt

- Theoretical computer sciences
- Program analysis (program analysis = abstraction interpretation + model checking)
- Program semantics
- Functional program language basics

What We Have Learnt

- Theoretical computer sciences
    - propositional logics (preliminary)
    - set theory (program semantics)
    - a basic abstract algebra (abstraction interpretation)
    - a basic automata theory (model checking)
    - a basic proof theory (program semantics)
- Program analysis (program analysis = abstraction interpretation + model checking)
- Program semantics
- Functional program language basics

What We Have Learnt

- Theoretical computer sciences
  - propositional logics (preliminary)
  - set theory (program semantics)
  - a basic abstract algebra (abstraction interpretation)
  - a basic automata theory (model checking)
  - a basic proof theory (program semantics)
- Program analysis (program analysis = abstraction interpretation + model checking)
  - model checking (Kripke structure, pushdown system) + (CTL, LTL)
  - abstraction interpretation
- Program semantics
- Functional program language basics

What We Have Learnt

- Theoretical computer sciences
    - propositional logics (preliminary)
    - set theory (program semantics)
    - a basic abstract algebra (abstraction interpretation)
    - a basic automata theory (model checking)
    - a basic proof theory (program semantics)
- Program analysis (program analysis = abstraction interpretation + model checking)
    - model checking (Kripke structure, pushdown system) + (CTL, LTL)
    - abstraction interpretation
- Program semantics
    - operational semantics
    - denotational semantics
    - axiomatic semantics
- Functional program language basics

What We Have Learnt

- Theoretical computer sciences
    - propositional logics (preliminary)
    - set theory (program semantics)
    - a basic abstract algebra (abstraction interpretation)
    - a basic automata theory (model checking)
    - a basic proof theory (program semantics)
- Program analysis (program analysis = abstraction interpretation + model checking)
    - model checking (Kripke structure, pushdown system) + (CTL, LTL)
    - abstraction interpretation
- Program semantics
    - operational semantics
    - denotational semantics
    - axiomatic semantics
- Functional program language basics
    - lambda calculus
    - simple type of lambda calculus

Theoretical Computer Sciences

Propositional Logic

Propositional Logic

Syntax and semantics

Propositional Logic

Syntax and semantics

Satisfiability and validity

Propositional Logic

Syntax and semantics

Satisfiability and validity

Normal form

Propositional Logic

Syntax and semantics

Satisfiability and validity

Normal form

- Tseitin's Encoding

Propositional Logic

Syntax and semantics

Satisfiability and validity

Normal form

- Tseitin's Encoding

Proof system

- Natural deduction
- Sequent calculus

Set Theory

Set Theory

Powerset, Indexed set, Big union, Big intersection

Set Theory

Powerset, Indexed set, Big union, Big intersection

Product, Disjoint union, Set difference, The axiom of foundation

Set Theory

Powerset, Indexed set, Big union, Big intersection

Product, Disjoint union, Set difference, The axiom of foundation

Binary relation, Partial function Total function

Set Theory

Powerset, Indexed set, Big union, Big intersection

Product, Disjoint union, Set difference, The axiom of foundation

Binary relation, Partial function Total function

Lambda notation, composition of functions, identity function, inverse, $1-1$ correspondence

Set Theory

Powerset, Indexed set, Big union, Big intersection

Product, Disjoint union, Set difference, The axiom of foundation

Binary relation, Partial function Total function

Lambda notation, composition of functions, identity function, inverse, $1-1$ correspondence

Direct image, inverse image, equivalence relation, equivalence class, transitive closure

Set Theory

Powerset, Indexed set, Big union, Big intersection

Product, Disjoint union, Set difference, The axiom of foundation

Binary relation, Partial function Total function

Lambda notation, composition of functions, identity function, inverse, $1-1$ correspondence

Direct image, inverse image, equivalence relation, equivalence class, transitive closure

Georg Cantor's Diagonal Argument

Abstract Algebra

Abstract Algebra

Preorder and partial order

- preorder: reflexivity, transitivity
- partial order: reflexivity, transitivity, antisymmetry

Abstract Algebra

Preorder and partial order

- preorder: reflexivity, transitivity
- partial order: reflexivity, transitivity, antisymmetry

well-quasi order

Abstract Algebra

Preorder and partial order

- preorder: reflexivity, transitivity
- partial order: reflexivity, transitivity, antisymmetry

well-quasi order

Preset, poset

Abstract Algebra

Preorder and partial order

- preorder: reflexivity, transitivity
- partial order: reflexivity, transitivity, antisymmetry

well-quasi order

Preset, poset

lub, glb

Abstract Algebra

Preorder and partial order

- preorder: reflexivity, transitivity
- partial order: reflexivity, transitivity, antisymmetry

well-quasi order

Preset, poset

lub, glb

group, ring, domain, lattice, complete lattice

Abstract Algebra

Preorder and partial order

- preorder: reflexivity, transitivity
- partial order: reflexivity, transitivity, antisymmetry

well-quasi order

Preset, poset

lub, glb

group, ring, domain, lattice, complete lattice

Tarski's fixpoint theorem

Automata Theory

Automata Theory

Finite automata: DFA, NFA

Infinite automata: Büchi automata

Pushdown automata and pushdown systems

Computation on automata

Automata Theory

Finite automata: DFA, NFA

Infinite automata: Büchi automata

Pushdown automata and pushdown systems

Computation on automata

Pumping lemma

Automata Theory

Finite automata: DFA, NFA

Infinite automata: Büchi automata

Pushdown automata and pushdown systems

Computation on automata

Pumping lemma

Antichain

Proof Theory

Proof Theory

Mathematical induction

Course-of-values induction

Proof Theory

Mathematical induction

Course-of-values induction

Structural induction

Proof Theory

Mathematical induction

Course-of-values induction

Structural induction

Well-founded induction, or Noetherian induction

Proof Theory

Mathematical induction

Course-of-values induction

Structural induction

Well-founded induction, or Noetherian induction

Induction on derivations

Proof Theory

Mathematical induction

Course-of-values induction

Structural induction

Well-founded induction, or Noetherian induction

Induction on derivations

Definition by induction

Program Analysis

Slogan

Program Analysis = Abstract Interpretation + Model Checking

Abstraction Interpretation

Specify an analysis in terms of the following data:

1. Preset D (abstract domain).
2. Monotone function F (abstract transfer function).
3. Widening operator ∇ (unless D is finite).
4. Narrowing operator Δ (optional).
5. Galois connection from C (concrete domain) to D.
6. Soundness of F wrt. E (concrete transfer function).

Then, the analysis terminates and is correct (sound).

Abstraction Interpretation

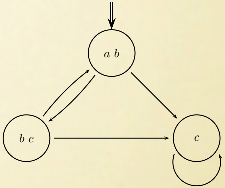
Patrick Cousot awarded John von Neumann Medal

Patrick Cousot is the recipient of the IEEE John von Neumann medal, given "for outstanding achievements in computer-related science and technology". The medal citation states that he is being recognized "for introducing abstract interpretation, a powerful framework for automatically calculating program properties with broad application to verification and optimization." Congratulations!

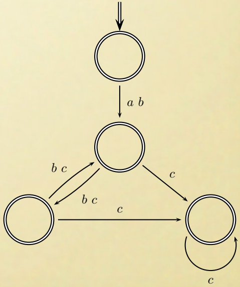
Model Checking: Model

- Kripke structure: $M = (S, S_0, R, L)$
  - S, finite set of state
  - $S_0 \subseteq S$, initial state
  - $R \subseteq S \times S$, transition relations
  - $L : S \rightarrow 2^{AP}$, status label function

    (AP: atomic propositions)

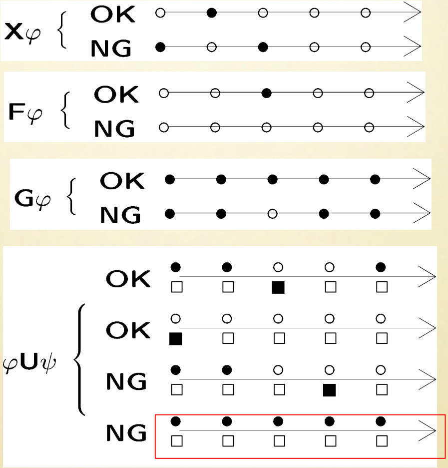


- Finite automata: $\mathcal{A} = (\Sigma, Q, Q_0, F, \delta)$
  - A, finite set of input alphabet
  - Q, finite set of control location
  - $Q_0 \subseteq Q$, initial control locations
  - $F \subseteq Q$, final control locations
  - $\delta \subseteq Q \times \Sigma \times Q$, transitions



Model Checking: Temporal Logic

- Next
- Finally
- Globally
- Until



● : φ, ■ : ψ, ○ : $\neg\varphi$, □ : $\neg\psi$

CTL Vs. LTL

- CTL: temporal operators must be immediately followed by path quantifiers.
  - e.g., $AF\varphi, EG\varphi, AXEG\varphi, EXA(\varphi U \psi)$
- LTL: path quantifiers are allowed only at the outermost position.
  - e.g., $AGF\varphi, EX(\varphi U \psi), A(F\varphi \vee G\psi)$
- Except for fairness, most properties are expressed in CTL $\cap$ LTL.

Decidability of CTL and LTL

Decidability of CTL and LTL

CTL: Polynomial algorithms

Decidability of CTL and LTL

CTL: Polynomial algorithms

LTL: PSPACE complete

Decidability of CTL and LTL

CTL: Polynomial algorithms

LTL: PSPACE complete

- by tableau
- by Büchi automata: on-the-fly model checking
- by SAT: bounded model checking

What We Did Not and Should Learn?

Symbolic model checking: ordered binary decision diagram (OBDD)

partial reduction: Spin

What We Did Not and Should Learn?

Symbolic model checking: ordered binary decision diagram (OBDD)

partial reduction: Spin

on-the-fly model checking

What We Did Not and Should Learn?

Symbolic model checking: ordered binary decision diagram (OBDD)

partial reduction: Spin

on-the-fly model checking

bounded model checking

What We Did Not and Should Learn?

Symbolic model checking: ordered binary decision diagram (OBDD)

partial reduction: Spin

on-the-fly model checking

bounded model checking

counter-example guided abstract refinement (CEGAR)

What We Did Not and Should Learn?

Symbolic model checking: ordered binary decision diagram (OBDD)

partial reduction: Spin

on-the-fly model checking

bounded model checking

counter-example guided abstract refinement (CEGAR)

Craig interpolation

What We Did Not and Should Learn?

Symbolic model checking: ordered binary decision diagram (OBDD)

partial reduction: Spin

on-the-fly model checking

bounded model checking

counter-example guided abstract refinement (CEGAR)

Craig interpolation

antichain

What We Did Not and Should Learn?

Symbolic model checking: ordered binary decision diagram (OBDD)

partial reduction: Spin

on-the-fly model checking

bounded model checking

counter-example guided abstract refinement (CEGAR)

Craig interpolation

antichain

Timed automata, well-structured transition systems (WSTS)

Program Semantics

operational semantics

denotational semantics

axiomatic semantics

What We Did Not and Should Learn?

Hoare logic, separation logic

What We Did Not and Should Learn?

Hoare logic, separation logic

Domain theory

What We Did Not and Should Learn?

Hoare logic, separation logic

Domain theory

Category theory basics

What We Did Not and Should Learn?

Hoare logic, separation logic

Domain theory

Category theory basics

Recursion theory

What We Did Not and Should Learn?

Hoare logic, separation logic

Domain theory

Category theory basics

Recursion theory

Gödel incompleteness

Functional Programming Languages

Lambda Calculus and Types

Lambda calculus syntax

β reduction and η reduction

Full β-reduction, normal order strategy, call by name, call by value

Programming in Lambda calculus

Church-Rosser Property

De Bruijn representation of Lambda calculus

Simple types of Lambda calculus

Progress and preservation

What We Did Not and Should Learn?

How to program in functional programming languages, such as SML, Ocaml, Haskell

Algorithms in functional programming languages

Structured types

Type checking and inferences

Normalization, References and Exceptions

Subtyping, recursive types

Higher order systems