Questions for oral exam, Logic in CS

Lucija Fekonja

7. julij 2024

1 Logic and type theory

1.1 Propositional logic

Lecture 1

1. Define a formula of propositional logic Φ.
   - Describe the rules.
2. How can we describe formulas?
3. What is the grammar for propositional logic formula?
4. Why are proof rules useful?
5. Define tautologies.
6. Write and explain proof rules for propositional logic
   - Conjunction introduction rules
   - Conjunction elimination rules
   - Disjunction introduction rules
   - Disjunction elimination rules
   - Implication introduction rules
   - Implication elimination rules
   - How would you explain Implication rules?
   - Negation introduction rules
   - Negation elimination rules
   - True rules
   - False rules
7. How can we write proofs?
8. EXAMPLE - Prove $p \wedge q \Rightarrow q \wedge p$ with a list.

9. EXAMPLE - Prove $p \wedge q \Rightarrow q \wedge p$ with a tree.
10. EXAMPLE - Prove $p \wedge (q \vee r) \Rightarrow (p \wedge q) \vee (p \wedge r)$ with a list.
11. EXAMPLE - Prove $p \wedge (q \vee r) \Rightarrow (p \wedge q) \vee (p \wedge r)$ with a tree.
12. How do we come to a contradiction?
13. Write a rule with which we can apply proof by contradiction?
14. EXAMPLE - Prove $\Phi \vee \neg\Phi$.
15. Simplify grammar.

1.2 Simple types and their proof rules and computation rules

Lecture 1

1. Give grammar of type theory with sum and product types.
   - What is necessery for a type theory grammar to have? (3)
2. Explain the analogy between types and formulas.
   - Product type introduction rules
   - Product type elimination rules
   - Analogy to what is product type?
   - Function type introduction rules
   - Function type elimination rules
   - Analogy to what is function type?
   - How would you explain function rules?
   - Sum type introduction rules
   - Sum type elimination rules
   - Rule for empty type
3. Name this analogy.
4. Give computation rules
   - Product type
   - Sum type
   - Function type
5. EXAMPLE – Derive a term of type $\sigma \times \tau \to \tau \times \sigma$.
6. EXAMPLE – Write a derivation of distributiviry to derive the term $\sigma \times (\tau + \Phi) \to (\sigma \times \tau) + (\sigma \times \Phi)$.

1.3 The universes Prop and Type

Lecture 2 professor's notes

1. With what can we replace the grammar for propositional logic?
2. Give rules in the Prop universe
   - False
   - Conjunction
   - Disjunction
   - Implication
3. How do we distinguish between judgements $\Phi : Prop$ and Φ?
4. With what can we replace the grammar for Types?
5. Give rules in the Type universe
   - Empty type
   - Product
   - Sum
   - Function
6. How do we distinguish between judgements $A : Type$ and $t : A$?

1.4 Quantifiers

Lecture 2

1. How do we also call predicate logic?
2. How do we get predicate logic form propositional logic?
3. How can we extend propositional logic? (3)
   - aka. What do we add to the grammar?
4. Give well formed rules for quantifiers
   - $\forall$
   - $\exists$
5. Give proof rules for quantifiers
   - $\forall i$
   - $\forall e$
   - $\exists i$
   - $\exists e$

6. What is the standard approach to predicate logic?
   - What do we choose?
   - EXAMPLE – Give examples.
   - How do we use it?
   - EXAMPLE – Give examples.
   - How do we use it's properties?
   - EXAMPLE – Give examples.
   - What do predicates express?
7. What is an atomic formula?
8. EXAMPLE – Give a simple example of atomic formula.
9. EXAMPLE – Give an example of using predicate logic using terms for $t, \cdot, 0, 1, <$.
   - What here are terms?
   - What here are atomic formulas?
   - What holds for variables?
   - What are free variables?
   - Does this formula have free variables?
   - Where is this formula true and where is it false?
10. What is the major limitation of traditional predicate logic?
11. Where is the problem from the perspective of programming?
12. How can we solve this problem?
13. Write an unified grammar.
14. With what can we replace grammar?
15. What does type theory give us?
16. What does predicate logic give us?
17. How can we combine the two?

1.5 Interesting types

Lecture 3

1. How do we call ”adding new types”?
2. What are formation rules used for?
3. EXAMPLE – Give examples of formation rules.
4. What do introduction rules give?
5. EXAMPLE – Give introduction rules for natural numbers.
6. What do elimination rules give?
7. Give four examples of type constructors / new types.
8. Formation rules for list type constructor.
9. How else can we write formation rule?
10. Introduction rules for list type constructor.
11. How else can we write introduction rule?
12. How do we construct a list?
13. Formation rules for vector type constructor.
14. How else can we write formation rule?
15. Introduction rules for vector type constructor.
16. How else can we write introduction rule?
17. How is $A \to B$ defined?
18. Formation rule for product type
19. Introduction rule for product type
20. Elimination rule for product type
21. Computation rule for product type
22. How is $A \times B$ defined?
23. Formation rule for dependent sum type
24. Introduction rule for dependent sum type
25. Elimination rules for dependent sum type
26. Computation rules for sum type

1.6 The BHK interpretation and the Curry-Howard correspondence

Lecture 3

1. Which elements do the following types have?
   - 0
   - $A \times B$
   - $A + B$
   - $A \to B$
   - $\pi x : A.B$
   - $\sum x : A.B$
2. What does BHK stand for?
3. What did they do?
4. Give proofs for the following terms:
   - $\perp$
   - $\Phi \wedge \Psi$
   - $\Phi \vee \Psi$
   - $\Phi \to \Psi$
   - $\forall x : A.\Phi$
   - $\exists x : A.\Phi$
5. What does Curry-Howard correspondance say?
6. For what kind of theory does it hold?
7. What can we use instead of:
   - $\perp$
   - $\Phi \wedge \Psi$
   - $\Phi \vee \Psi$
   - $\Phi \to \Psi$
   - $\forall x : A.\Phi$
   - $\exists x : A.\Phi$
8. Who developed this approach?
9. When?
10. What approach does Lean take?

11. Describe this approach.

12. Define subuniverse

13. What is a subterminal property?

14. What do we also need here?

15. How do we change the formation rule for product type?

16. What is Scott-Prawitz interpretation of logic?

17. How did they define the following terms?

    - $\perp$
    - Which equality do we have using this definition?
    - $\Phi \wedge \Psi$
    - $\Phi \vee \Psi$
    - $\Phi \to \Psi$
    - $\forall x : A.\Phi$
    - $\exists x : A.\Phi$

18. Explain $\Phi \wedge \Psi$ of Scott-Prawitz interpretation

    - If we want to derive introduction rule
    - If we want to derive elimination rule

1.7 Natural numbers, lists and vectors as inductive types

Lecture 4 **Nat**

1. Write out induction on natural numbers

2. Define the eliminator for natural numbers

3. What do we need do define for the eliminator? (name it)

4. Give computation rules for natural numbers

    - How do we use the recursor on 0?
    - How do we use the recursor on succ n?

5. EXAMPLE – Proposition (fun), result, function

6. How do we define a sum of two natural numbers?

7. Give an improved version of n+m

8. EXERCISE – Derive $succ0 + succ0$

Lists

1. Give constructors for lists (2)
2. Give eliminator for lists
3. Give computation rules for lists (2)
4. EXERCISE - Define append on lists using R_{list}.

Vectors

1. Give constructors for vectors (2)
2. Give eliminator for vectors
3. Give computation rules for vectors (2)

Propositional equality

1. Give formation rule
2. Define constructur rfl
3. Define eliminator
4. Give computation rule
5. What is the intuition for propositional equality?

Judgemental equality

1. What do we define?
2. Give formulation rule for judgemental equality.
3. Give substitution rule for judgemental equality.

1.8 Tantologies and satisfiability

(Lecture 5)

1. Determine grammar of propositional logic.
2. What is a formula?
3. What are it's arguments?
4. What can we determine with a formula?
5. EXAMPLE - Formula in propositional Logic.
6. Define a valid tantology.
7. Define a satisfiable formula.

8. What is such b called?
9. Define equivalence.
10. How can we establish that a formula is a tantology? (2)
11. In what time do they work?
12. What theorems do we have for natural deduction?
13. Give soundness theorem.
14. Give completeness thorem.
15. What is the size of the truth table.
16. When is a formula a tantology based on the truth table?
17. What can we also determine with a truth table?
18. How?

2 SAT problem

2.1 Normal forms for formulas

Lecture 5

1. Define the negation normal form.
2. Define the grammar of NNF.
3. What does deMorgan say about NNF?
4. How can we rewrite a formula for it to be written in NNF? (5)
5. In what time can we compute NNF from the original?

2.2 Conjunctive and normal form

Lecture 5

1. Define a literal
2. Define a clause
3. What is a clause of length 0?
4. Define conjunctive normal form
5. What is a CNF of length 0?
6. Define a co-clause

7. Define a disjunctive normal form.
8. What proposition do we make on CNF and DNF?
9. Give an recursive algorithm for computing Φ^{CNF}. (6)
    - What do we assume for Φ?
10. In what time can we compute CNF and DNF?
11. ————-???————

2.3 SAT problem

Lecture 5

1. What is the input?
2. What is the output?
3. What is the related problem to SAT?
4. What is it's input?
5. What is it's output?
6. How are this two problems related?
7. In what time can Φ be solved?
    - In what form does Φ have to be written for that to be the case?
8. Exactly when is a DNF satisfiable?
9. What is the open question?
10. What would be proved if this it true?
11. When is SAT solving sufficient?
12. What transformation do we define?
13. What is the imageof this mapping?
14. What property does this transformation have?
15. In what time can it be computed?
16. For what is SAT solving useful?
17. What is a checkable solution called?
18. EXAMPLE - Give an example of a problem that can be solved with SAT
    - What is a 3-coloring?

- What is the question?
- How do we solve it?
- What do we define?
- How does the formula look like?
- What holds for every edge?
- When does a graph have a 3-colouring?

19. Explain DPLL algorithm

2.4 Examples

Lecture 6

1. We change the algorithm for SAT solver for it to be informative. How? • Input? • Output?
2. In what time can SAT work?
3. EXAMPLE - $\Phi = \neg((x \vee \neg y) \wedge \neg(x \wedge \neg(y \wedge z)))$
   - Create a tree
   - Substitute
   - Create a list of clauses
4. EXAMPLE - Sudoku
   - How do we represent a problem?
   - How many variables do we have?
   - What is the goal?
   - What holds for square 11 because there has to be to least one number in every square?
   - How many such clauses exist?
   - What holds for square 11 because there has to be to most one number in every square?
   - How many such clauses exist?
   - How many clauses exist?
   - How many clauses exist for a 4x4 sudoku?

3 Predicate logic and bounded model checking

3.1 Signatures, terms and formulas

Lecture 7

1. EXAMPLE - Give example properties that could be used in computer systems (4)

2. Define syntax

3. Define semantics

4. EXAMPLE - Axioms

   - Example axiom - squares
   - Where is this true?
   - Where is it false?
   - Example axiom - inverse
   - Where is this axiom true?
   - Where is it false?
   - What inthe past three examples is new vocabulary?

5. How do we specify vocabulary?

6. Define signature

7. What does every predicate have?

8. What does each function have?

9. What are (P, F) in CS example?

10. What are (P, F) in axioms example?

11. What does signature determine? (2)

12. What do terms express?

13. What do formulas express?

14. What is always a term?

15. How can we construct terms?

16. EX - What are terms in CS example?

17. EX - What are terms in axiom example?

18. What are always formulas?

19. How can we construct formulas from terms?
20. How can we construct formulas from formulas?
21. How else can we construct formulas?
22. What here are atomic formulas?
23. How are the semantics given?

3.2 Free and bound variables, sentences, substitution

1. Interpret $\exists y.x = y \times y$ in N
   - What does the formula express?
   - What is a free variable?
   - What is a bound variable?
   - In what scope?
2. What kind of variables are not in axiom exercise (exercise 2)?
3. What is such a formula called?

3.3 Structures, the satisfaction relation, satisfiability, validity, the statement of Gödel's completeness theorem

1. Formally define structure
   - M
   - p
   - f
2. Give natural interpretation of $(\leq, +, \cdot, 0, 1)$
3. What do we get with such interpretation?
4. What is each such interpretation?
5. How do we interpert a term in a structure?
6. What is σ?
7. What does it map?
8. Where is a variable mapped?
9. Where is a function mapped?
10. How are formulas interpreted?
11. How do we write an interpretation?

12. How do we read it?
13. When are the following interpretations true?
    - $m \models_\sigma P(t_1, \ldots t_n)$
    - $m \models_\sigma t_1 = t_2$
    - $m \models_\sigma \Phi \to \Psi$
    - $m \models_\sigma \Phi \land \Psi$
    - $m \models_\sigma \Phi \lor \Psi$
    - $m \models_\sigma \neg\Phi$
    - $m \models_\sigma \forall x.\Phi$
    - $m \models_\sigma \exists x.\Phi$
14. What here are atomic formulas?
15. When is a truth value of $m \models_\Sigma \Phi$ independent of Σ?
16. How do we write this?
17. Define a valid sentance
18. Define a satisfiable sentance
19. Give examples of valid formulas (2)
20. Give examples of satisfiable formulas
21. How are validity and satisfiability conected?
22. THEOREM - Turing / Church theorem
23. How do we interpret this theorem?
24. THEOREM - Gödels completeness theorem
25. Give the problem where Gödels theorem is used
    - What is the input?
    - What is the output?
    - What is this algorithm called?
    - Is it efficient?
    - Where is it used?

3.4 The computational status of validity and satisfiability

Lecture 7

1. What algorithm do we get with Gödels completeness theorem and the proof-checking algotrithm?
    - Input?
    - Output?
    - What is the practical result?
    - What are the theoretical results? (2)
    - What algorithm therefore exists? (because of validity)
    - What is the input?
    - What is the output?
    - What practical consequence does it have?
    - What algorithm therefore exists? (because of satisfiability)
    - What is the input?
    - What is the output?

3.5 Finite satisfiability and bounded satisfiability

Lecture 8

1. Create 6 axioms with predicates sever, client, connected
2. Define a model
3. What is a model checking question?
4. What algorithm do we need?
5. What do we compute/return in the following cases?
    - $m \models_\sigma P(t_1, \ldots t_n)$
    - $m \models_\sigma t_1 = t_2$
    - $m \models_\sigma \Phi \wedge \Psi$
    - $m \models_\sigma \neg\Phi$
    - $m \models_\sigma \exists x.\Phi$
6. What kind of problem is this?
7. In what time does algorithm work?
8. How is this run time called?
9. Can we make it polynomial?

10. How?
11. Depentent on what?
12. What axioms can we add (in the example from question 1)?
13. What are we now interested in?
14. Can you ask this question in a different way? More formally
15. Algorithm for what do we need?
    - Input?
    - Output?
16. Is the problem decidable?
17. What algorithm does exist?
18. How would a naive algorithm work?
19. Is it good? Why?
20. Name the algorithm that is more practical
    - What is it's input?
    - What question does it answer?
    - What is the output?

3.6 Reducing bounded satisfiability to SAT solving

1. What is the main idea of bounded satisfiability thorem?
2. Which stages does the algorithm have?
3. Describe stage 1
    - What is the input?
    - What do we add?
    - What do we build?
    - Using what?
    - What is the idea?
4. After we have a sentence in predicate logic, what do we define?
5. Where does this mapping map...
    - Predicate
    - Negation

- Conjinction
- Existance

6. FACT - When is Φ satisfied with a structure?
7. How do we get rid of function symbols in the propositional formula?
   - With what do we replacee functions?
   - What axioms do we add? (2)
8. How do we treat equality =?
   - What axioms do we add? (4)

4 Linear-time temporal logic

4.1 Transition system, LTL syntax, runs and the LTL satisfaction relation

Lecture 9

1. Define temporal logic.
2. Define linear time.
3. For what is LTL good?
4. Give an example of something where LTL would be good to use.
5. What system did we define? Name it.
   - How did we represent it?
   - What components did we define?
   - Draw a diagram.
6. What properties can a LTL system have?
   - 1 - name it
   - 1 - what is the definition of the property?
   - 1 - does our example system have it?
   - 2 - what is the definition of the property?
   - 2 - does our example system have it?
   - 2 - give an counterexample.
   - 3 - name it.
   - 3 - what is the definition of the property?
   - 4 - name it

- 4 - what is the definition of the property?

7. Define a transition system for LTL.
   - S
   - $\rightarrow$
   - property
   - L
   - anything else?
8. How can we model a deadlock?
9. What syntax for formulas did we define for temporal logic? (8)
10. Define every symbol.
11. Define a run.
12. How do we also call a run?
13. What notation can we create from a standard notation of a run?
14. What relation did we define in temporal logic?
15. What is it's notation?
16. How do we read the notation?
17. When is $\Pi \models \ldots$ true? (give equality)
    - p
    - $\Phi \wedge \Psi$
    - $\Phi \vee \Psi$
    - $\Phi \rightarrow \Psi$
    - $\neg\Phi$
    - $\bot$
    - $\top$
    - $X\Phi$
    - $G\Phi$
    - $F\Phi$
    - $\Phi U \Psi$
    - $\Phi W \Psi$
    - $\Phi R \Psi$

4.2 Examples of fairness constraints and how they are used

Lecture 9

1. What is a LTL model checking?
    - Input?
    - Output?
2. EXAMPLE - What does the algorithm return if we input $G((t_1 \to Fc_1) \wedge (t_2 \to Fc_2))$?
3. EXAMPLE - Slightly redraw the diagram you drew earlier
    - What does the algorithm return?
    - When would the original model return true?
    - Define this constraint
    - With what can we model check?
    - What does the algorithm return?

4.3 NBAs (and GNBAs), their associated omega languages and the emptiness condition

1. We defined automats. What are they called?
2. EXAMPLE - Give a simple example
    - Draw a diagram
    - Give an example of a word this automat would accept.
    - When does this automat accept a ω-word?
    - Define a ω-word.
    - Give some more examples of words that are / are not accepted.
    - Define ω-language of the automations.
    - Formally write ω-language.
3. EXAMPLE - Give another example of a NBA.
    - Draw a diagram.
    - What is ω-language in this example?
4. EXAMPLE - Give a third example of a NBA.
    - Draw a diagram.
    - What is ω-language in this example?

5. Define an NBA over an alphabet
6. When is a ω-word accepted in an NBA over an alphabet?
7. PROPOSITION - When is a ω-language of NBA non empty?
8. What are we testing here?
9. Define ω-regular
10. THEOREM - How can we construct new ω-regular sets? (4)
11. Define a general NBA.
12. When is a word accepted?
13. EXAMPLE - of a GNBA.
    - Draw a diagram.
    - What is the alphabet?
    - What is the ω-language?
14. PROPOSITION - How are NBA and GNBA connected? *ucilnica*
15. THEOREM - What holds for all GNBA?

4.4 Trace language of an LTL formula

Ucilnica lecture 10

1. What do we have to consider?
2. What holds for the systems we define?
3. What do we have to suppose?
4. What follows for all LTL formulas?
5. On what does $\pi \models \Phi$ depend?
6. What do we therefore define?
7. How can we also look at the trace?
8. What does a formula Φ determine?
    - What is it called?
    - How do we "notate it"?
9. Formally write out $Trace(\Phi)$.
10. How do we connect GNBA and Trace?

11. What assumption do we make on Φ?
12. How can we express $F\phi$ with these connectives? *zapiski*
13. How can we express $G\phi$ with these connectives? *zapiski*
14. Define a completed subformula of Φ. (6 cases)
15. How is $CS(aUb)$ defined?
16. Define an elementary subset. (5 cases)
17. EXAMPLE - What are elementary subsets of $CS(aUb)$? (5 cases)
18. Draw a diagram
19. What do we have for every $\Phi U \Phi'$.
20. Define $S \rightarrow^A S'$. (5 cases)
21. THEOREM - What is a subset of A_Φ.

4.5 The model checking problem for LTL

Ucilnica lecture 10

1. What is the input?
2. What is the question?
3. STEP 1
    - What do we construct?
    - To what is it converted?
    - To what does an accepted run in converted structure correspond?
4. STEP 2
    - What do we construct?
    - What are the accepted runs in this structure?
5. How do we construct a product NBA?
    - Define m
    - Define A
    - Define a set of states
6. How do we get an NBA from GNBA?
7. STEP 3
    - What do we test?

- How do we test this?
- When do we output YES?
- When do we output NO?

8. EXAMPLE - Illustrate the model checking on $m, s_0 \models \neg(aUb)$

5 Hoare logic (logic for verifing programs)

Lecture 11 + ucilnica

5.1 Proof rules for partial correctness

1. Where is it used?
2. Why is hoare logic important?
3. In what syles can we write proof?
4. Give proof rules in tree style
   - Partial while
   - Composition
   - Assignment
   - Skip
   - Conditional
   - Consequence
   - What is the side condition for the consequence rule?
5. Give tableaux rules
   - Assignment
   - Skip
   - Partial while
   - What is the precondition for while?
   - Implied
   - If statement
   - What are the preconditions for if?

5.2 Statement of soundness

1. Define $\models_{par} \{\phi\}C\{\psi\}$
2. Define $\vdash_{par} \{\phi\}C\{\psi\}$
3. Soundness theorem
4. How is the soundness theorem proved?
5. LEMMA - Proving preservation property of partial while
6. Proof
   - How do we proof this?
   - What has to be satisfied?
   - Write proof for $n = 0$
   - Write proof for $n - 1 \to n$

5.3 Total correctness

1. Proof rule for total while in tree style
   - What is z_0?
   - What is η?
   - What is E?
2. Proof rule of total while in tableaux style
3. Define $\models_{tot} \{\phi\}C\{\psi\}$
4. Define $\vdash_{tot} \{\phi\}C\{\psi\}$
5. Soundness theorem
6. LEMMA - Proving preservation property of partial while
7. Proof
   - How do we proof this?
   - What has to be satisfied?
   - Write proof for $n = 0$
   - Write proof for $n - 1 \to n$

5.4 Completeness for partial correctness

1. Completeness theorem
2. How is completeness theorem also called?
3. Why is it called so?
4. What is Gödels theorem called?
5. Define notation $wp(C, \psi)$
6. Name 3 lemmas that use wp
7. LEMMA - expressive completeness
8. LEMMA - Weakest precondition
9. LEMMA - Sequencing

5.5 Proof of completeness

1. How is the proof structured?
2. What case of the proof did we show?
3. What induction hypothesis do we have for this case?
4. What did we suppose?
5. What do we have to proove?
   - 1
   - 2
   - 3
6. What follows if these
7. Prove 1
8. Prove 2
9. Prove 3