Logik für Informatiker
Logic for computer scientists

Till Mossakowski, Lutz Schröder

WiSe 2011/12

Overview

- Why is logic needed in computer science?
- Overview of the course
- The LPL book and software
- “Scheinkriterien”

Why is logic needed in computer science?

- formal specification and verification
- databases, WWW, artificial intelligence
- algorithms & complexity
- metatheory
- (semi-)automated theorem proving
- programming languages

Why is logic needed in computer science?

- formal specification and verification
- databases, WWW, artificial intelligence
- algorithms & complexity
- metatheory
- (semi-)automated theorem proving
- programming languages

Why is logic needed in computer science?

- formal specification and verification
- databases, WWW, artificial intelligence
- algorithms & complexity
- metatheory
- (semi-)automated theorem proving
- programming languages

Why is logic needed in computer science?

- formal specification and verification
- databases, WWW, artificial intelligence
- algorithms & complexity
- metatheory
- (semi-)automated theorem proving
- programming languages

Why is logic needed in computer science?

- formal specification and verification
- databases, WWW, artificial intelligence
- algorithms & complexity
- metatheory
- (semi-)automated theorem proving
- programming languages

Why is logic needed in computer science?

- formal specification and verification
- databases, WWW, artificial intelligence
- algorithms & complexity
- metatheory
- (semi-)automated theorem proving
- programming languages

Why is logic needed in computer science?

- formal specification and verification
- databases, WWW, artificial intelligence
- algorithms & complexity
- metatheory
- (semi-)automated theorem proving
- programming languages

Formal specification and verification

- formal software and hardware development
- verification of existing software and hardware
- generation of test cases
- protocol verification, security (modal and temporal logics)
- properties of telephone systems
- Example: Pentium 4 arithmetic completely specified and verified with higher-order logic!
- Example: NASA uses logic for testing software

Formal specification and verification

- formal software and hardware development
- verification of existing software and hardware
- generation of test cases
- protocol verification, security (modal and temporal logics)
- properties of telephone systems
- Example: Pentium 4 arithmetic completely specified and verified with higher-order logic!
- Example: NASA uses logic for testing software

Formal specification and verification

- formal software and hardware development
- verification of existing software and hardware
- generation of test cases
- protocol verification, security (modal and temporal logics)
- properties of telephone systems
- Example: Pentium 4 arithmetic completely specified and verified with higher-order logic!
- Example: NASA uses logic for testing software

Formal specification and verification

- formal software and hardware development
- verification of existing software and hardware
- generation of test cases
- protocol verification, security (modal and temporal logics)
- properties of telephone systems
- Example: Pentium 4 arithmetic completely specified and verified with higher-order logic!
- Example: NASA uses logic for testing software

Formal specification and verification

- formal software and hardware development
- verification of existing software and hardware
- generation of test cases
- protocol verification, security (modal and temporal logics)
- properties of telephone systems
- Example: Pentium 4 arithmetic completely specified and verified with higher-order logic!
- Example: NASA uses logic for testing software

Formal specification and verification

- formal software and hardware development
- verification of existing software and hardware
- generation of test cases
- protocol verification, security (modal and temporal logics)
- properties of telephone systems
- Example: Pentium 4 arithmetic completely specified and verified with higher-order logic!
- Example: NASA uses logic for testing software

Formal specification and verification

- formal software and hardware development
- verification of existing software and hardware
- generation of test cases
- protocol verification, security (modal and temporal logics)
- properties of telephone systems
- Example: Pentium 4 arithmetic completely specified and verified with higher-order logic!
- Example: NASA uses logic for testing software

Formal specification and verification

- formal software and hardware development
- verification of existing software and hardware
- generation of test cases
- protocol verification, security (modal and temporal logics)
- properties of telephone systems
- Example: Pentium 4 arithmetic completely specified and verified with higher-order logic!
- Example: NASA uses logic for testing software

databases, WWW, artificial intelligence

- queries for web search; database queries (SQL)
- ontologies and semantic web
- expert systems
- linguistics
- Example: CYC is a very large knowledge base containing over 1.5 Million "facts, rules-of-thumb and heuristics for reasoning about the objects and events of everyday life"
  —the CYC inference engine uses first-order logic!

databases, WWW, artificial intelligence

- queries for web search; database queries (SQL)
- ontologies and semantic web
- expert systems
- linguistics
- Example: CYC is a very large knowledge base containing over 1.5 Million "facts, rules-of-thumb and heuristics for reasoning about the objects and events of everyday life"
  —the CYC inference engine uses first-order logic!

databases, WWW, artificial intelligence

- queries for web search; database queries (SQL)
- ontologies and semantic web
- expert systems
- linguistics
- Example: CYC is a very large knowledge base containing over 1.5 Million "facts, rules-of-thumb and heuristics for reasoning about the objects and events of everyday life"
  —the CYC inference engine uses first-order logic!

databases, WWW, artificial intelligence

- queries for web search; database queries (SQL)
- ontologies and semantic web
- expert systems
- linguistics
- Example: CYC is a very large knowledge base containing over 1.5 Million "facts, rules-of-thumb and heuristics for reasoning about the objects and events of everyday life"
  —the CYC inference engine uses first-order logic!

databases, WWW, artificial intelligence

- queries for web search; database queries (SQL)
- ontologies and semantic web
- expert systems
- linguistics
- Example: CYC is a very large knowledge base containing over 1.5 Million "facts, rules-of-thumb and heuristics for reasoning about the objects and events of everyday life"
  —the CYC inference engine uses first-order logic!

databases, WWW, artificial intelligence

- queries for web search; database queries (SQL)
- ontologies and semantic web
- expert systems
- linguistics
- Example: CYC is a very large knowledge base containing over 1.5 Million "facts, rules-of-thumb and heuristics for reasoning about the objects and events of everyday life"
  —the CYC inference engine uses first-order logic!

Algorithms & complexity

- if a graph property can be stated in monadic second-order logic, there is an efficient algorithm for it
- complexity classes can be characterized by classes of logical formulas
- Example: the proof of NL=Co-NL was based on this — the hope is to push this further towards P=?NP

Algorithms & complexity

- if a graph property can be stated in monadic second-order logic, there is an efficient algorithm for it
- complexity classes can be characterized by classes of logical formulas
- Example: the proof of NL=Co-NL was based on this — the hope is to push this further towards P=?NP

Algorithms & complexity

- if a graph property can be stated in monadic second-order logic, there is an efficient algorithm for it
- complexity classes can be characterized by classes of logical formulas
- Example: the proof of NL=Co-NL was based on this — the hope is to push this further towards P=?NP

Algorithms & complexity

- if a graph property can be stated in monadic second-order logic, there is an efficient algorithm for it
- complexity classes can be characterized by classes of logical formulas
- Example: the proof of NL=Co-NL was based on this — the hope is to push this further towards P=?NP

Metatheory

- set theory has been formalized in first-order logic
  — this serves as a foundations for all of mathematics and theoretical computer science
- Gödel's completeness theorem for first-order logic: semantics can be captured by formal proofs
  — even by machine-driven proofs!
- Gödel's incompleteness theorem
  for first-order logic + induction:
  some essential pieces of mathematics and theoretical computer science cannot be captured by formal systems!

Metatheory

- set theory has been formalized in first-order logic
  — this serves as a foundations for all of mathematics and theoretical computer science
- Gödel's completeness theorem for first-order logic: semantics can be captured by formal proofs
  — even by machine-driven proofs!
- Gödel's incompleteness theorem
  for first-order logic + induction:
  some essential pieces of mathematics and theoretical computer science cannot be captured by formal systems!

Metatheory

- set theory has been formalized in first-order logic
  — this serves as a foundations for all of mathematics and theoretical computer science
- Gödel's completeness theorem for first-order logic: semantics can be captured by formal proofs
  — even by machine-driven proofs!
- Gödel's incompleteness theorem
  for first-order logic + induction:
  some essential pieces of mathematics and theoretical computer science cannot be captured by formal systems!

Metatheory

- set theory has been formalized in first-order logic
  — this serves as a foundations for all of mathematics and theoretical computer science
- Gödel's completeness theorem for first-order logic: semantics can be captured by formal proofs
  — even by machine-driven proofs!
- Gödel's incompleteness theorem
  for first-order logic + induction:
  some essential pieces of mathematics and theoretical computer science cannot be captured by formal systems!

(Semi-)automated theorem proving

- logical properties of finite state machines can be automatically checked (model checkers)
- more complex systems need semi-automated proving
- verification of proofs is easy and fully automatic
- Example: some theorem about Boolean algebras has been found by a computer
- Example: several math text books have been verified with a semi-automatic prover
  (and small but inessential errors have been found)

(Semi-)automated theorem proving

- logical properties of finite state machines can be automatically checked (model checkers)
- more complex systems need semi-automated proving
- verification of proofs is easy and fully automatic
- Example: some theorem about Boolean algebras has been found by a computer
- Example: several math text books have been verified with a semi-automatic prover
  (and small but inessential errors have been found)

(Semi-)automated theorem proving

- logical properties of finite state machines can be automatically checked (model checkers)
- more complex systems need semi-automated proving
- verification of proofs is easy and fully automatic
- Example: some theorem about Boolean algebras has been found by a computer
- Example: several math text books have been verified with a semi-automatic prover
  (and small but inessential errors have been found)

(Semi-)automated theorem proving

- logical properties of finite state machines can be automatically checked (model checkers)
- more complex systems need semi-automated proving
- verification of proofs is easy and fully automatic
- Example: some theorem about Boolean algebras has been found by a computer
- Example: several math text books have been verified with a semi-automatic prover
  (and small but inessential errors have been found)

(Semi-)automated theorem proving

- logical properties of finite state machines can be automatically checked (model checkers)
- more complex systems need semi-automated proving
- verification of proofs is easy and fully automatic
- Example: some theorem about Boolean algebras has been found by a computer
- Example: several math text books have been verified with a semi-automatic prover
  (and small but inessential errors have been found)

(Semi-)automated theorem proving

- logical properties of finite state machines can be automatically checked (model checkers)
- more complex systems need semi-automated proving
- verification of proofs is easy and fully automatic
- Example: some theorem about Boolean algebras has been found by a computer
- Example: several math text books have been verified with a semi-automatic prover
  (and small but inessential errors have been found)

Programming languages

- Many programming languages use logical and, or, not
- Prolog = programming in logic
- concentrates on what instead of how
- involves non-deterministic search
- used for applications in linguistics and artificial intelligence

Programming languages

- Many programming languages use logical and, or, not
- Prolog = programming in logic
- concentrates on what instead of how
- involves non-deterministic search
- used for applications in linguistics and artificial intelligence

Programming languages

- Many programming languages use logical and, or, not
- Prolog = programming in logic
- concentrates on what instead of how
- involves non-deterministic search
- used for applications in linguistics and artificial intelligence

Programming languages

- Many programming languages use logical and, or, not
- Prolog = programming in logic
- concentrates on what instead of how
- involves non-deterministic search
- used for applications in linguistics and artificial intelligence

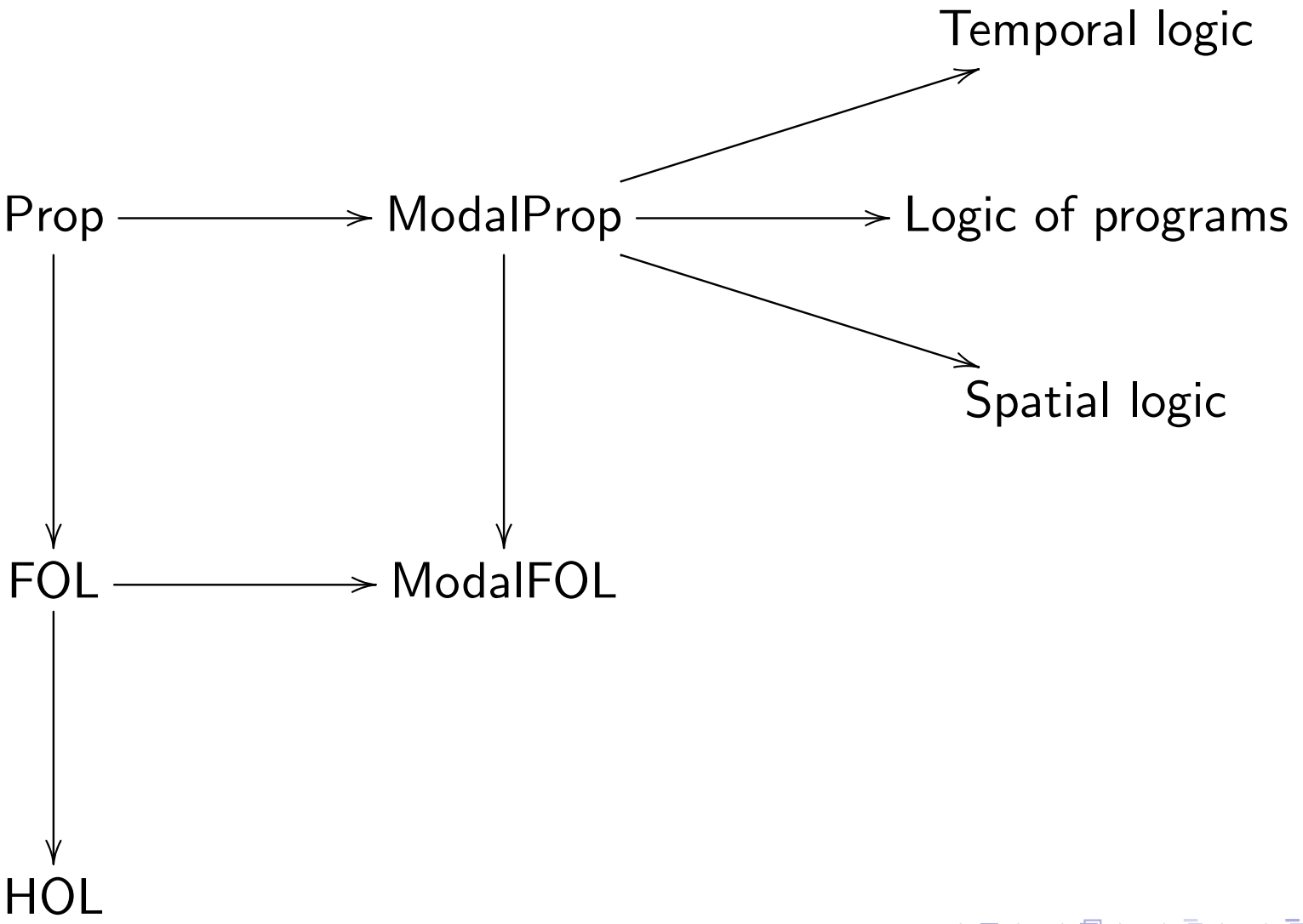
Programming languages

- Many programming languages use logical and, or, not
- Prolog = programming in logic
- concentrates on what instead of how
- involves non-deterministic search
- used for applications in linguistics and artificial intelligence

Programming languages

- Many programming languages use logical and, or, not
- Prolog = programming in logic
- concentrates on what instead of how
- involves non-deterministic search
- used for applications in linguistics and artificial intelligence

Landscape of logics



- propositional consequence
- Hintikka games
- propositional proofs
- resolution
- (semi-)automatic proving: SPASS, Isabelle
- first-order quantifiers
- first-order consequence

Language, proof and logic

LPL book	detailed introduction into first-order logic with many exercises
Boole	construct truth tables
Tarski's world	evaluate logical formulas within a blocks world
Fitch	construct proofs
Grinder	gives automatic feedback to your solutions $\rightarrow$ requires purchase of the CD (ca. 13 EUR) or the book (ca. 25 EUR, with CD)

Language, proof and logic

LPL book	detailed introduction into first-order logic with many exercises
Boole	construct truth tables
Tarski's world	evaluate logical formulas within a blocks world
Fitch	construct proofs
Grinder	gives automatic feedback to your solutions $\rightarrow$ requires purchase of the CD (ca. 13 EUR) or the book (ca. 25 EUR, with CD)

Rooms

- Tuesday 14:00 - 16:00 MZH 1400
- Thursday 16:00 - 18:00 MZH 1470
- Exercises (bring your Laptops with you!) within the course
- No lecture on Thursday, November 11
- Web: `www.informatik.uni-bremen.de/agbkb/lehre/ws11-12/Logik/`

Course Criteria

- successful solution of 10 exercises from 7 different chapters, with deadlines as given in the course
  - to be found in the LPL book
  - *but:* only those listed on the website, marked with grades
  - grade is average of 10 best solutions, but only as good as the best fitch solution
  - groups of 1-3 students (10/20/30 exercises, same grade for all)
  - submitted to the Grinder or to me (depending on the exercise)
- and: presentation of solutions to the class, or oral exam ("Fachgespräch")

Language, proof and logic

- for working with the Grinder, *each* student/group needs an *own new* CD
- important: *register* at logic.informatik.uni-bremen.de under the *same* e-mail as for the Grinder
- try easy exercises first, to reach the minimum of 10 (later on, you can improve: only the 10 best solutions count)
- *only* exercises with a successful report (by the Grinder or us) count
- the Grinder is always right (but some old versions of Fitch are buggy)