Logic week 1

Lecture 1

Syllabus of the logic part of this module 本模块的逻辑部分的教学节

- Propositional logic
  命题逻辑
  - syntax syntax
    语法
  - proofs(natural deduction & sequent calculus)
    证明（自然推导和后续演算）
  - semantics, truth tables 语义、真值表
  - satisfiability
    可满足性
- First order logic (predicate calculus)
  一阶逻辑（谓词微算）
  - syntax
    语法
  - proofs (natural deduction & sequent calculus)
    证明（自然推导和后续演算）
  - semantics
    语义
- Theorem proving
  定理证明
  - propositional & predicate logic
    命题和谓词逻辑
  - data types, induction & recursion
    数据类型、归纳法和递归
  - numbers
    数
- Constructive logic 构造性逻辑
  - classical vs. constructive logic
    经典逻辑与构造逻辑
  - lambda-calculus
    λ演算
  - realizability
    可实现性
  - simply-typed lambda calculus
    简单类型λ演算

Basic concepts

- ***Proposition***
  命题
  - A **proposition** is a sentence which states a fact 一个命题是陈述一个事实的句子
    i.e. a statement that can (in principle) be true or false
    (原则上)可以是真的或假的陈述
  - Example
    - Birmingham is north of London
      proposition, and true
    - Please mind the gap
      not a proposition!
- ***Arguments***
  参数

    - An **argument** is a list of propositions
      一个**论证**是一个命题的列表
        - The last of which is called the conclusion
          最后一个叫做结论
        - and the others are called premises
          其他的叫做前提
    - Example
        - Premise 1: If there is smoke, then there is a fire
          Premise 2: There is no fire
          Conclusion: Therefore, there is no smoke

- ***Validity of Arguments***
  参数的有效性
    - An argument is **valid** if (and only if), whenever the premises are true, then so is the conclusion
      一个论点是有效的如果(和当且仅当),每当前提为真,那么结论就是结论
    - Example
        - Premise 1: If there is smoke, then there is a fire
          Premise 2: There is no fire
          Conclusion: Therefore, there is no smoke

Conclusion

- What did we cover today?
    - what and why logic
    - organization of the logic part of the module
    - basic logic concepts

Lecture 2

Symbolic Logics 符号逻辑

- Symbolic logics are **formal languages** that allow conducting logical reasoning through the **manipulation of symbols**
  符号逻辑是一种形式语言，它允许通过对符号的操作进行逻辑推理
- Example:
    - Propositional logic
    - Predicate logic
    - Higher-order logic

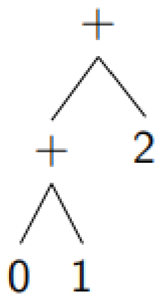
Grammars - BNFs

- Two important aspect of a language are:
    - its **syntax** describing the well-formed sequences of symbols denoting objects of the language;
      它的语法描述表示语言对象的符号序列的语法；
    - and its **semantics** assigning meaning to those symbols
      以及它赋予这些符号意义的语义
- The syntax of a language is defined through a **grammar**.
  一种语言的语法是通过一种语法来定义的
- **Backus Naur Form(BNF)** allows defining **context-free** grammars.
    - lhs ::“ rhs$_1$ | ¨ ¨ ¨ | rhs$_n$
      Meaning: this rule means that the left-hand-side lhs (a non-terminal symbol) can expand to any of the forms rhs1 to rhsn on the right-hand-side.
      该规则意味着左侧的lhs（一个非终端符号）可以扩展到右侧的任何形式的rhs$_1$到rhs$_n$。
    - Each **rhs$_i$** is a sequence of non-terminal and terminal symbols
      每个rhs$_i$都是一个非终端符号和终端符号的序列。
    - The **arity** of a terminal symbol is the number of arguments it takes
      终端符号的统一性是它所需要的参数的数量。
    - The **Fixity** of a terminal symbol is the place where it occurs w.r.t.
      终端符号的固定度是它发生w.r.t.的地方其论点：

    - **infix** if it occurs in-between its arguments
      插缀，如果发生在参数之间
    - **prefix** if it occurs before 前缀，如果它出现在前缀之前
    - **postfix** if it occurs after
      后缀，如果它发生在之后

Grammars - abstract syntax trees 抽象语法树

- An expression derived from a BNF grammar can then be seen as a tree, called an **abstract syntax tree**
  从BNF语法派生的表达式可以被视为一个树，称为抽象语法树。
    - Example:
      exp ::“ num | exp + exp | exp * exp



Grammars - associativity 结合性

- Define the **associativity** of the terminal symbols to avoid ambiguities
  定义终端符号的**结合性**以避免二义性
- Example: 0 + 1 + 2
    - left associativity: (0 + 1) + 2
    - right associativity: 0 + (1 + 2)

Grammars - precedence 优先权

- Define the **precedence** of the terminal symbols to avoid ambiguities
  定义终端符号的**优先级**以避免歧义
- example: 0 + 1 * 2
    - * has higher precedence: 0 + (1 * 2)
    - + has higher precedence: (0 + 1) * 2

Grammars - associativity, precedence, parentheses 联想性、优先级，括号

To avoid ambiguities:
为了避免含糊不清:

- define the associativity of symbols
  定义符号的结合性
- define the precedence between symbols
  定义符号之间的优先级
- use parentheses to avoid ambiguities or for clarity
  使用圆括号以避免歧义或清晰

Parentheses are sometimes necessary: 有时需要使用圆括号：

- using left associativity 0 + 1 + 2 stands for (0 + 1) + 2
- we need parentheses to express 0 + (1 + 2)

(Meta)variables （元）变量

metavariables or **schematic variables** act as placeholders for any element derivable from a given grammar rule.
元变量或示意图变量作为可从给定的语法规则派生出来的任何元素的占位符。

Axiom schemata 公理模式

- Axioms can be obtained by instantiating the variable exp with any arithmetic expression. This is called an axiom schemata.
  公理可以通过用任意算术表达式实例化变量exp来获得。这被称为公理图式。

Substitution 替换

- A **substitution** is a mapping, that maps multivariables to arithmetic expressions
  替换是一种映射，它将多个变量映射到算术表达式
- The **substitution operation** is the operation that replaces all occurrences of the keys by the corresponding values
  替换操作是用相应的值替换键的所有出现的操作
- Example:
  (exp + 0 = exp)[exp\1]
  returns 1 + 0 = 1
- v[s] = v, if v is not a key of s v不在s中 所以对结果不影响
- v[s] = e, if s maps v to e v在s中 所以对结果有影响

Conclusion

- What did we cover today?
  - A formal language such as a symbolic logic has a syntax captured by a grammar
  - (Meta)variables are used to capture collections of axioms(as axiom schemata) of symbolic logics
  - Substitution is used to derive instances of axiom schemata