That's all Folks!
Final exam

- Tuesday 12/13 2pm–5pm in this room
  2 double-sided pages of notes

- Cumulative, but will focus a bit more on recent topics
  nope: Turing machines • proofs about languages • Prolog
1.1 Syntax of the Predicate Calculus

Definition. The syntax of the predicate calculus (\(\mathcal{PC}\)) consists of symbols and formulas as follows:

Symbols
parentheses: (, )

sentential connectives: ¬, ∨, ∧, →, ↔

quantifiers: ∀, ∃

\(SC\) letters (sentential letters): A, B, ⋯, Z, and any of these letters with a positive Arabic numeral subscripts.

predicate symbols: An \(n\)-ary predicate is an uppercase letter, \(A, ⋯, Z\), with the numeral \(n\) as a superscript, where \(n\) denotes the arity of the predicate and \(0 < n\). These uppercase letters may also have numerical subscripts. Note: We will usually omit the superscript when we know the arity of a predicate.

individual constants: lowercase letters a, ⋯, r, with or without numerical subscripts.

individual variables: lowercase letters s, ⋯, z, with or without numerical subscripts.

Formulas
The set of all predicate calculus (\(\mathcal{PC}\)) formulas is defined recursively, beginning with the atomic formulas.

Atomic Formula:
Any single \(SC\) letter, or an \(n\)-ary predicate followed by exactly \(n\) symbols, each of which is either an individual constant or a variable.

Formula:
Any atomic formula, or any expression (finitely long string of symbols) that is obtainable by use of the following predicate calculus construction rules (PCCR):
Prolog is syntactic sugar for the predicate calculus.

*Prolog is syntactic sugar for the predicate calculus.*

**Syntax of the Predicate Calculus**

- **Predicate (.),**
  - Unsorted constants: a, b, c, ...
- Sorted constants: i, e
- Sorted function symbols: f, g, ...
  - with arguments: f(a, b, c)

**Formulas**

The set of all predicate calculus (PC) formulas is defined recursively, beginning with the atomic formulas:

- **Atomic Formulas**
  - Any single PC letter, or no more predicate followed by exactly one argument, each of which is either an individual constant or a variable.

**Formulas**

Any formulas, or any expansion through logical operations that is derivable by one of the following predicate calculus rules (PC rules):

- **The Semantics of Predicate Logic as a Programming Language**

  M. H. Van Emden and R. A. Kowalski

  University of Edinburgh, Edinburgh, Scotland

  **Abstract**

  Sentences in first-order predicate logic can be profitably interpreted as programs. In this paper, the operational and denotational semantics of predicate logic programs are defined, and the connections with the proof theory and model theory of logic are investigated. It is concluded that operational semantics is a part of proof theory and that denotational semantics is a special case of model-theoretic semantics.

  **Key Words and Phrases**: predicate logic as a programming language, semantics of programming languages, resolution theorem proving, operational versus denotational semantics, PC-resolution, program characterization.

  **Categories**: F.2.2, 5.2.1, 5.2.4

1. Introduction

Predicate logic plays an important role in many formal models of computer programs [3, 4, 17]. Here we are concerned with the interpretation of predicate logic as a programming language [5, 16]. The Prolog system (for PROgramming in LOGic), based upon the procedural interpretation, has been used for several ambitious programming tasks, including English-language question answering [5, 18].
Hmmm  ≡  Racket  ≡  Prolog

\[ 0 \text{ jumpn } 0 \]

\[ \lambda \text{ Calculus } \equiv \text{ Prolog } \]

Turing Machine  ≡  \lambda \text{ Calculus}

\[ 0 \text{ jumpp } 0 \]

\[ ( (\lambda (x) (x x)) (\lambda (x) (x x)) ) \]

\[ p : - p \]
1. Language influences thought.

2. Programs $\equiv$ Data
   and self-reference is powerful

3. Computers are powerful.
   they change our lives for good and bad
Language influences thought.
Programs ≡ Data
and self-reference is powerful.
Computers are powerful.
they change our lives for good and bad
Commentary: Facebook’s Algorithm vs. Democracy

By Cathy O’Neil on Wed, 07 Dec 2016

Over the last several years, Facebook has been participating—unintentionally—in the erosion of democracy.

The social network may feel like a modern town square, but thanks to its tangle of algorithms, it’s nothing like the public forums of the past. The company determines, according to its interests and those of its shareholders, what we see and learn on its social network. The result has been a loss of focus on critical national issues, an erosion of civil disagreement, and a threat to democracy itself.

Facebook is just one part—though a large part—of the Big Data economy, one
Margaret H. Hamilton
Margaret H. Hamilton led the team that created the on-board flight software for NASA’s Apollo command modules and lunar modules. A mathematician and computer scientist who started her own software company, Hamilton contributed to concepts of asynchronous software, priority scheduling and priority displays, and human-in-the-loop decision capability, which set the foundation for modern, ultra-reliable software design and engineering.

Grace Hopper (posthumous)
Rear Admiral Grace Hopper, known as “Amazing Grace” and “the first lady of software,” was at the forefront of computers and programming development from the 1940s through the 1980s. Hopper’s work helped make coding languages more practical and accessible, and she created the first compiler, which translates source code from one language into another. She taught mathematics as an associate professor at Vassar College before joining the United States Naval Reserve as a lieutenant (junior grade) during World War II, where she became one of the first programmers of the Harvard Mark I computer and began her lifelong leadership role in the field of computer science.

Katherine G. Johnson
Katherine G. Johnson is a pioneer in American space history. A NASA mathematician, Johnson’s computations have influenced every major space program from Mercury through the Shuttle program. Johnson was hired as a research mathematician at the Langley Research Center with the National Advisory Committee for Aeronautics (NACA), the agency that preceded NASA, after they opened hiring to African-Americans and women.