Introduction
& Review

York University
Department of Computer Science and Engineering

Overview

- Why this course?
- Programming Language Paradigms
- Brief review of Logic
  - Propositional logic
  - Predicate logic

- Ref:
  - R.W. Sebesta, Concepts of Programming languages-7th edition, Pearson Education, 2006.
  - G. Tourlakis, Mathematical Logic, John Wiley & Sons, 2008.
  - Prof. Stachniak's class notes

Why this course?

- From the undergraduate calendar:

  This course covers **functional and logic programming**. Together with the students' background on procedural and object-oriented programming, the course allows them to compare the development of programs in these different types of languages.

- Reasons for studying concepts of programming languages [Sebesta]:
  1. Increased capacity to express ideas
  2. Improved background for choosing appropriate languages
  3. Increased ability to learn new languages
  4. Better understanding of the significance of implementation
  5. Overall advancement of computing

Programming Language Paradigms

- (1) Imperative programming
  - Semantics (what the program does) is state based; involves variables and assignments
  - Computation viewed as state transition process
  - Categories:
    - Procedural, e.g. C, Pascal, Turing
    - Visual, e.g. Visual Basic: code can be dragged & dropped
    - Object Oriented, e.g. Java
    - Other non-structured

Imperative: Of the nature of or expressing a command; commanding (dictionary.reference.com)

Programming Language Paradigms-cont.

- (2) Declarative Programming
  - Focus is on logic (WHAT) rather than control (HOW)
  - Categories:
    - Logic Programming: Computation is a reasoning process, e.g. Prolog
    - Functional Programming: Computation is the evaluation of a function, e.g. Lisp, Scheme, ...
    - Constrained Languages: Computation is viewed as constraint satisfaction problem, e.g. Prolog (R)

Declarative: Serving to declare, make known, or explain (dictionary.reference.com)

Ref.: Sebesta, R.W. Concepts of Programming Languages, 7th edition

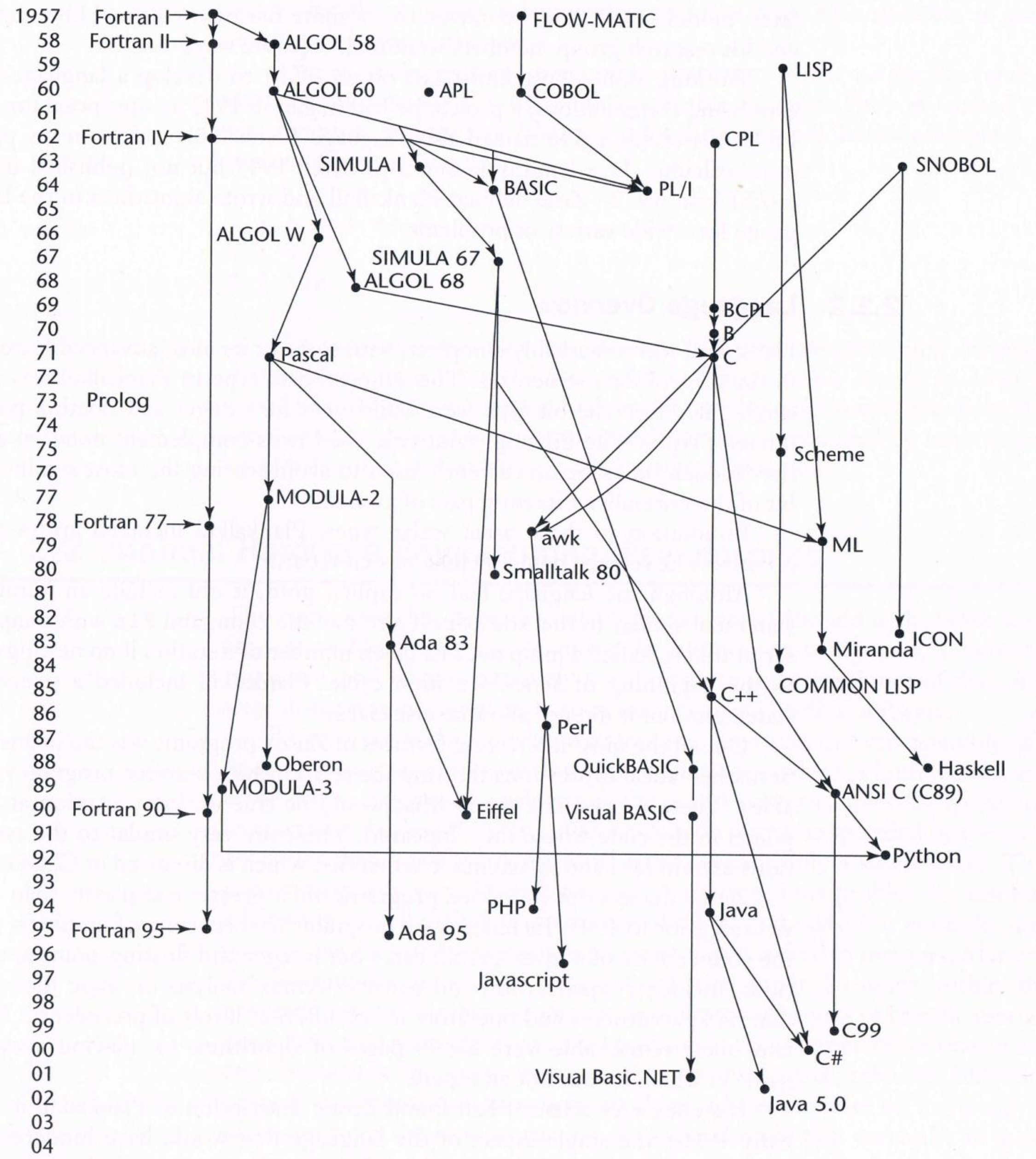


Figure 2.1 Genealogy of common high-level programming languages

Programming Language Paradigms-cont.

- Level of language
  - Low level
    - has a world view close to that of the computer
  - High level
    - has a world view closer to that of the specification (describing the problem to be solved, or the structure of the system to be presented)

- Evaluation Criteria:
  - Readability, Writability, Reliability, Cost

- Design and evaluation depend on the domain and the problem to be solved

PART I- LOGIC PROGRAMMING

Why Logic Programming?

- View of the world imposed by a language

  A programming language tends to impose a certain view of the world on its users. Logic Programming is based on Logic and reasoning.

- Semantics of the programming languages

  To program with the constructs of a language requires thinking in terms of the semantics of those constructs. Logic programming requires thinking in terms of facts and rules.

Logic Programming

- Based on *first order predicate logic*
- A programmer *describes* with formulas of predicate logic
- A *mechanical problem solver* makes inferences from these formulas

Propositional Logic (review)

- Alphabet
  - Variables, e.g. p, q, r, ..., p_1, ..., p', ...
  - Constants: T and $\perp$ (or F)
  - Connectives: $\{\neg, \wedge, \vee, \rightarrow, \equiv\}$
    - or {~, &, #, ->, <-> in some books}
  - Brackets: (and)

- Well-formed-formula (wff)
  - All variables and constants are wffs.
  - If A and B are wffs, then the following are also wffs.

    $(\neg A),\ \ (A \wedge B),\ \ (A \vee B),\ \ (A \rightarrow B),\ \ (A \equiv B)$

  - Priority of connectives, and rules for removing brackets

Propositional Logic (cont.)

- Semantics and truth tables
  - true (1) and false (0)
  - State: possible assignment to variables
- Tautology
  - A formula A is a <u>tautology</u> if v(A)=1 (true) in all possible states
  - Example: $(p \vee \neg p)$
- Satisfiable / consistent
  - A formula A is <u>satisfiable</u> iff there is at least one state v where v(A)=1 (true). Examples: $p,\ (p \wedge q),\ (p \rightarrow q)$
  - A set of formulae X is <u>satisfiable</u> (or <u>consistent</u>) iff there is at least one state v where for every formula A in X, v(A)=1. Example: $\{p, (p \wedge q), (p \rightarrow q)\}$

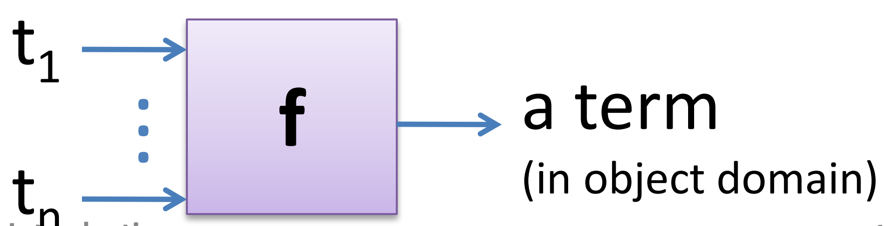
Propositional Logic (cont.)

- Unsatisfiable / inconsistent / contradiction
  - A formula A is unsatisfiable (or a contradiction) iff no state v exists where v(A)=1, in other words for all possible states v(A)=0 (false). Example: $(p \wedge \neg p)$
  - Note if A is a tautology, (¬A) is a contradiction.
  - A set of formulae is unsatisfiable (or inconsistent) iff for all possible states v at least one formula A in the set is false, i.e. v(A)=0. Example:

    $$\{p, p \wedge q, p \rightarrow \neg q\}$$

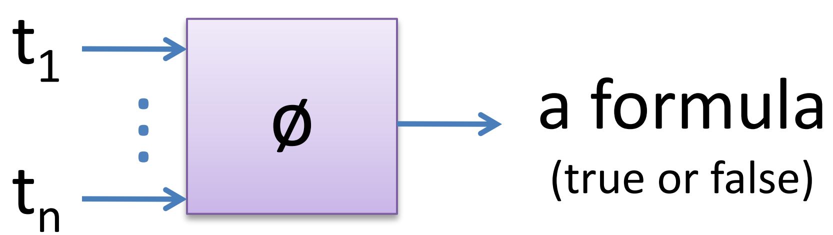
Predicate Logic (review)

- Alphabet
  - Alphabet of propositional logic
  - Object variables, e.g. x, y, z, ..., x_1, ..., x', ...
  - Object constants, e.g. a, b, c, ...
  - Object equality symbol =
  - Quantifier symbols $\forall$ (and $\exists$)
  - and some functions & predicates
- Term
  - An object variable or constant, e.g. x, a
  - A function f of n arguments, where each argument is a term, e.g. $f(t_1, t_2, ...t_n)$



Predicate Logic (cont.)

- Atomic formula
  - A Boolean variable or constant
  - The string t = s, where t and s are terms
  - A predicate ø of n arguments where each argument is a term , e.g. $ø(t_1, t_2, \ldots t_n)$



- Well-formed formula
  - Any atomic formula
  - If A and B are wffs, then the following are also wffs.

  $(\neg A)$, $(A \wedge B)$, $(A \vee B)$, $(A \rightarrow B)$, $(A \equiv B)$, $((\forall x)A)$, $((\exists x)A)$

Examples

- Numbers
  - Object constants: 1, 2, 3, ...
  - Functions: +, -, *, /, ...
  - Predicates: >, <, ...
  - Examples of wffs: $>(x,y) \rightarrow >(+(x,1),y)$

    Or the familiar notation: $x > y \rightarrow x+1 > y$

    Another example: $x! = z \rightarrow (x+1)! = (x+1) * z$

- Sets
  - Object constants: {1}, {2,3},...
  - Functions: $\bigcup$, $\bigcap$,...
  - Predicates: $\subset$, $\subseteq$,...
  - A wff: $(x \bigcap y) \subseteq (x \bigcup y)$

More Examples

- Our world
  - Object variables: X, Y, ...
    - upper case in PROLOG
  - Constants such as: john, mary, book, fish, flowers, ...
    - Note lower case in PROLOG
  - Functions: distance(point1, X), wife(john)
  - Predicates: owns(book, john), likes(mary, flowers), ...
  - true and false in PROLOG
    - Relative to PROLOG's knowledge of the world
    - False whenever it cannot find it in its database of facts (and rules)