UNIVERSITÄT
BERN

Programming Languages

1. Introduction

Prof. O. Nierstrasz
Spring Semester 2010

Programming Languages

Lecturer:	Oscar Nierstrasz
Assistants:	Toon Verwaest, Camillo Bruni
WWW:	http://scg.unibe.ch/teaching/pl

Roadmap

- Course Schedule
- Programming Paradigms
- A Quick Tour of Programming Language History

Roadmap

- **Course Schedule**
- Programming Paradigms
- A Quick Tour of Programming Language History

Sources

Text:

- Kenneth C. Louden, *Programming Languages: Principles and Practice*, PWS Publishing (Boston), 1993.

Other Sources:

- Paul Hudak, "Conception, Evolution, and Application of Functional Programming Languages," ACM Computing Surveys 21/3, 1989, pp 359-411.
- Clocksin and Mellish, *Programming in Prolog*, Springer Verlag, 1987.

Schedule

1. Introduction
2. Stack-based programming
3. Scheme (guest lecture)
4. Functional programming
5. Types and polymorphism
6. Lambda calculus
7. Fixed points
8. Programming language semantics
9. Objects and types
10. Logic programming
11. Applications of logic programming
12. Visual programming
13. *Final exam*

Roadmap

- Course Schedule
- **Programming Paradigms**
- A Quick Tour of Programming Language History

What is a Programming Language?

- A formal language for describing computation?
- A “user interface” to a computer?
- Syntax + semantics?
- Compiler, or interpreter, or translator?
- A tool to support a programming paradigm?

> *A programming language is a notational system for describing computation in a machine-readable and human-readable form.*
>
> *— Louden*

What is a Programming Language? (II)

The thesis of this course:

A programming language is a tool for developing executable models for a class of problem domains.

Themes Addressed in this Course

Paradigms

How do different language paradigms support problem-solving?

Foundations

What are the foundations of programming languages?

Semantics

How can we understand the semantics of programming languages?

Generations of Programming Languages

1GL: machine codes

2GL: symbolic assemblers

3GL: (machine-independent) imperative languages (FORTRAN, Pascal, C ...)

4GL: domain specific application generators

5GL: AI languages …

Each generation is at a higher level of abstraction

How do Programming Languages Differ?

Common Constructs:

- basic data types (numbers, etc.); variables; expressions; statements; keywords; control constructs; procedures; comments; errors ...

Uncommon Constructs:

- type declarations; special types (strings, arrays, matrices, ...); sequential execution; concurrency constructs; packages/modules; objects; general functions; generics; modifiable state; ...

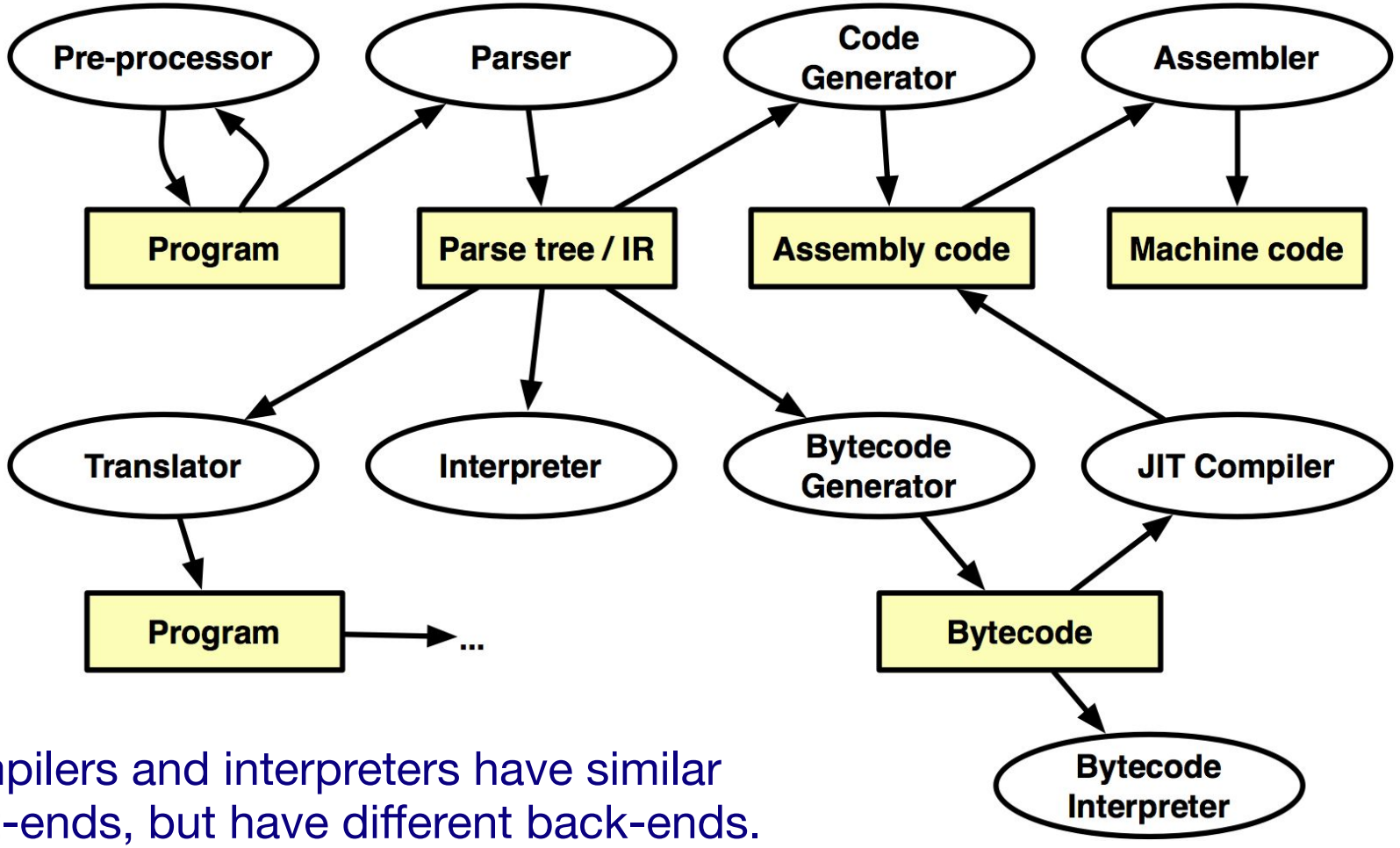
Programming Paradigms

A programming language is a problem-solving tool.

Imperative style:	program = algorithms + data *good for decomposition*
Functional style:	program = functions ° functions *good for reasoning*
Logic programming style:	program = facts + rules *good for searching*
Object-oriented style:	program = objects + messages *good for modeling(!)*

Other styles and paradigms: blackboard, pipes and filters, constraints, lists, ...

Compilers and Interpreters



Compilers and interpreters have similar front-ends, but have different back-ends.

Roadmap

- Course Schedule
- Programming Paradigms
- **A Quick Tour of Programming Language History**

A Brief Chronology

Early 1950s		*“order codes” (primitive assemblers)*
1957	FORTRAN	*the first high-level programming language*
1958	ALGOL	*the first modern, imperative language*
1960	LISP, COBOL	*Interactive programming; business programming*
1962	APL, SIMULA	*the birth of OOP (SIMULA)*
1964	BASIC, PL/I	
1966	ISWIM	*first modern functional language (a proposal)*
1970	Prolog	*logic programming is born*
1972	C	*the systems programming language*
1975	Pascal, Scheme	*two teaching languages*
1978	CSP	*Concurrency matures*
1978	FP	*Backus’ proposal*
1983	Smalltalk-80, Ada	*OOP is reinvented*
1984	Standard ML	*FP becomes mainstream (?)*
1986	C++, Eiffel	*OOP is reinvented (again)*
1988	CLOS, Oberon, Mathematica	
1990	Haskell	*FP is reinvented*
1990s	Perl, Python, Ruby, JavaScript	*Scripting languages become mainstream*
1995	Java	*OOP is reinvented for the internet*
2000	C#	

Fortran

History

- *John Backus (1953) sought to write programs in conventional mathematical notation, and generate code comparable to good assembly programs.*
- No language design effort (made it up as they went along)
- Most effort spent on code generation and optimization
- FORTRAN I released April 1957; working by April 1958
- The current standard is FORTRAN 2003
  (FORTRAN 2008 is work in progress)

Fortran ...

Innovations

- Symbolic notation for subroutines and functions
- Assignments to variables of complex expressions
- DO loops
- Comments
- Input/output formats
- Machine-independence

Successes

- Easy to learn; high level
- Promoted by IBM; addressed large user base
- (scientific computing)

“Hello World” in FORTRAN

```
    PROGRAM HELLO
    DO 10, I=1,10
    PRINT *,'Hello
World'
10  CONTINUE
    STOP
    END
```

All examples from the ACM "Hello World" project:
www2.latech.edu/~acm/HelloWorld.shtml

ALGOL 60

History

- *Committee of PL experts formed in 1955 to design universal, machine-independent, algorithmic language*
- First version (ALGOL 58) never implemented; criticisms led to ALGOL 60

Innovations

- BNF (Backus-Naur Form) introduced to define syntax (led to syntax-directed compilers)
- First block-structured language; variables with local scope
- Structured control statements
- Recursive procedures
- Variable size arrays

Successes

- Highly influenced design of other PLs but never displaced FORTRAN

“Hello World” in BEALGOL

```
BEGIN
FILE F (KIND=REMOTE);
EBCDIC ARRAY E [0:11];
REPLACE E BY "HELLO WORLD!";
WHILE TRUE DO
    BEGIN
    WRITE (F, *, E);
    END;
END.
```

COBOL

History

- *Designed by committee of US computer manufacturers*
- Targeted business applications
- Intended to be readable by managers (!)

Innovations

- Separate descriptions of environment, data, and processes

Successes

- Adopted as de facto standard by US DOD
- Stable standard for 25 years
- Still *the most widely used PL* for business applications (!)

“Hello World” in COBOL

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.     HELLOWORLD.
000300 DATE-WRITTEN.   02/05/96        21:04.
000400* AUTHOR BRIAN COLLINS
000500 ENVIRONMENT DIVISION.
000600 CONFIGURATION SECTION.
000700 SOURCE-COMPUTER. RM-COBOL.
000800 OBJECT-COMPUTER. RM-COBOL.
001000 DATA DIVISION.
001100 FILE SECTION.
100000 PROCEDURE DIVISION.
100200 MAIN-LOGIC SECTION.
100300 BEGIN.
100400       DISPLAY " " LINE 1 POSITION 1 ERASE EOS.
100500       DISPLAY "HELLO, WORLD." LINE 15 POSITION
10.
100600       STOP RUN.
100700 MAIN-LOGIC-EXIT.
100800       EXIT.
```

PL/1

History

- *Designed by committee of IBM and users (early 1960s)*
- Intended as (large) general-purpose language for broad classes of applications

Innovations

- Support for concurrency (but not synchronization)
- Exception-handling on conditions

Successes

- Achieved both run-time efficiency and flexibility (at expense of complexity)
- First “complete” general purpose language

“Hello World” in PL/1

```
HELLO:    PROCEDURE OPTIONS (MAIN);

             /* A PROGRAM TO OUTPUT HELLO WORLD
*/
             FLAG = 0;

LOOP:     DO WHILE (FLAG = 0);
             PUT SKIP DATA('HELLO WORLD!');
             END LOOP;

END HELLO;
```

Functional Languages

ISWIM (If you See What I Mean)

- Peter Landin (1966) — paper proposal

FP

- John Backus (1978) — Turing award lecture

ML

- Edinburgh
- initially designed as meta-language for theorem proving
- Hindley-Milner *type inference*
- “non-pure” functional language (with assignments/side effects)

Miranda, Haskell

- “pure” functional languages with *“lazy evaluation”*

“Hello World” in Functional Languages

SML

```
print("hello world!\n");
```

Haskell

```
hello() = print "Hello World"
```

Prolog

History

- Originated at U. Marseilles (early 1970s), and compilers developed at Marseilles and Edinburgh (mid to late 1970s)

Innovations

- Theorem proving paradigm
- Programs as sets of clauses: facts, rules and questions
- Computation by “unification”

Successes

- Prototypical logic programming language
- Used in Japanese Fifth Generation Initiative

“Hello World” in Prolog

```
hello :- printstring("HELLO WORLD!!!!").

printstring([]).
printstring([H|T]) :- put(H), printstring(T).
```

Object-Oriented Languages

History

- ***Simula*** was developed by Nygaard and Dahl (early 1960s) in Oslo as a language for simulation programming, by adding *classes and inheritance* to ALGOL 60

```
Begin
    while 1 = 1 do begin
        outtext ("Hello
World!");
        outimage;
    end;
End;
```

- ***Smalltalk*** was developed by Xerox PARC (early 1970s) to drive graphic workstations

```
Transcript show:'Hello World';cr
```

Object-Oriented Languages

Innovations

- *Encapsulation* of data and operations (contrast ADTs)
- *Inheritance* to share behaviour and interfaces

Successes

- Smalltalk project pioneered OO user interfaces
- Large commercial impact since mid 1980s
- Countless new languages: C++, Objective C, Eiffel, Beta, Oberon, Self, Perl 5, Python, Java, Ada 95 ...

Interactive Languages

Made possible by advent of time-sharing systems (early 1960s through mid 1970s).

BASIC

- Developed at Dartmouth College in mid 1960s
- Minimal; easy to learn
- Incorporated basic O/S commands (NEW, LIST, DELETE, RUN, SAVE)

```
10 print "Hello World!"
20 goto 10
```

...

Interactive Languages ...

APL

- Developed by Ken Iverson for concise description of numerical algorithms
- Large, non-standard alphabet (52 characters in addition to alphanumerics)
- Primitive objects are arrays (lists, tables or matrices)
- Operator-driven (power comes from composing array operators)
- No operator precedence (statements parsed right to left)

```
'HELLO WORLD'
```

Special-Purpose Languages

SNOBOL

- First successful string manipulation language
- Influenced design of text editors more than other PLs
- String operations: pattern-matching and substitution
- Arrays and associative arrays (tables)
- Variable-length strings

...

```
    OUTPUT = 'Hello
World!'
END
```

Symbolic Languages ...

Lisp

- Performs computations on symbolic expressions
- Symbolic expressions are represented as *lists*
- Small set of constructor/selector operations to create and manipulate lists
- Recursive rather than iterative control
- *No distinction between data and programs*
- First PL to implement storage management by garbage collection
- Affinity with lambda calculus

```
(DEFUN HELLO-WORLD ()
       (PRINT (LIST 'HELLO
'WORLD)))
```

4GLs

"Problem-oriented" languages

- PLs for "non-programmers"
- Very High Level (VHL) languages for specific problem domains

Classes of 4GLs (no clear boundaries)

- Report Program Generator (RPG)
- Application generators
- Query languages
- Decision-support languages

Successes

- Highly popular, but generally ad hoc

“Hello World” in RPG

```
H
FSCREEN O F 80 80
CRT
C                                       EXCPT
OSCREEN E 1
O                                                    12 'HELLO
WORLD!'
```

“Hello World” in SQL

```
CREATE TABLE HELLO (HELLO CHAR(12))
UPDATE HELLO
    SET HELLO = 'HELLO WORLD!'
SELECT * FROM HELLO
```

Scripting Languages

History

Countless “shell languages” and “command languages” for operating systems and configurable applications

- ***Unix shell*** (ca. 1971) developed as user shell and scripting tool

```
echo "Hello, World!"
```

- ***HyperTalk*** (1987) was developed at Apple to script HyperCard stacks

```
on OpenStack
     show message box
     put "Hello World!" into message
box
end OpenStack
```

- ***TCL*** (1990) developed as embedding language and scripting language for X windows applications (via Tk)

```
puts "Hello World "
```

- ***Perl*** (~1990) became de facto web scripting language

```
print "Hello, World!\n";
```

Scripting Languages ...

Innovations

- Pipes and filters (Unix shell)
- Generalized embedding/command languages (TCL)

Successes

- Unix Shell, awk, emacs, HyperTalk, AppleTalk, TCL, Python, Perl, VisualBasic ...

The future?

- Dynamic languages
  - very active
- Domain-specific languages
  - very active
- Visual languages
  - many developments, but still immature
- Modeling languages
  - emerging from UML and MDE …

What you should know!

- *What, exactly, is a programming language?*
- *How do compilers and interpreters differ?*
- *Why was FORTRAN developed?*
- *What were the main achievements of ALGOL 60?*
- *Why do we call C a “Third Generation Language”?*
- *What is a “Fourth Generation Language”?*

Can you answer these questions?

- *Why are there so many programming languages?*
- *Why are FORTRAN and COBOL still important programming languages?*
- *Which language should you use to implement a spelling checker?*
- *A filter to translate upper-to-lower case?*
- *A theorem prover?*
- *An address database?*
- *An expert system?*
- *A game server for initiating chess games on the internet?*
- *A user interface for a network chess client?*

License

http://creativecommons.org/licenses/by-sa/3.0/