Principles of Programming Languages
Lecture 2

Wael Aboulsaadat

wael@cs.toronto.edu

http://portal.utoronto.ca/

Acknowledgment: parts of these slides are based on material by Diane Horton & Eric Joanis @ UoT

References: Scheme by Dybvig
Concepts of PL by Sebesta
Prog. in Prolog by Clocksin and Mellish
PL Concepts and Constructs by Sethi
ML for the Working Prog. By Paulson
PL Pragmatics by Scott

Administrative: Waivers

- **Course waivers**

- **CGPA waivers**
  - **Check with your department**

Introduction to Logic Programming

Logic Programming (LP)

- **Evolution:**

  Problem → Algorithm → Assembly Code → Machine Code

  |-----------*Assembler*------------------|

  |----*Imperative/functional Compiler/Interpreter*--|

  |------------------*Logic Language Compiler/Interpreter*---------|

- **E.g.:**
  - Find X and Y such that 3 X + 2 Y = 1 and X – Y = 4
  - Retrieve the telephone number of the person whose name is Tom Smith
  - The value of X equals the value of Y + 3

- **Why LP?**
  - We can understand the meaning without knowing the “state” of the program
  - A lot easier to say *what*, but not *how*.
  - Direct manipulation of symbolic structures gives us it’s power.

- **Popular LP languages:** *Prolog, SQL, Datalog*

LP: introduction – cont’d

- **LP Characteristics:**
  - Not based on state modifications
  - Not procedural in nature
  - Does not have control flow (*as we are used to thinking of it*)

So, what does it have?

- **A program in a logic programming language consists of a set of *declarations* related together using *predicate calculus*.**

- **An algorithm in a LP language = logic + control**
  - Logic: programmer provides the “logic” which is what the program does.
  - Control: language run-time system provides the control (!)

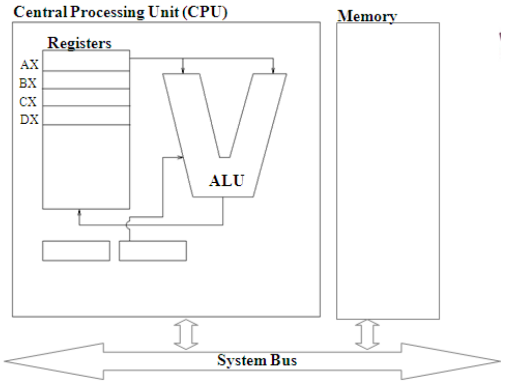
LP: Library Software Case Study

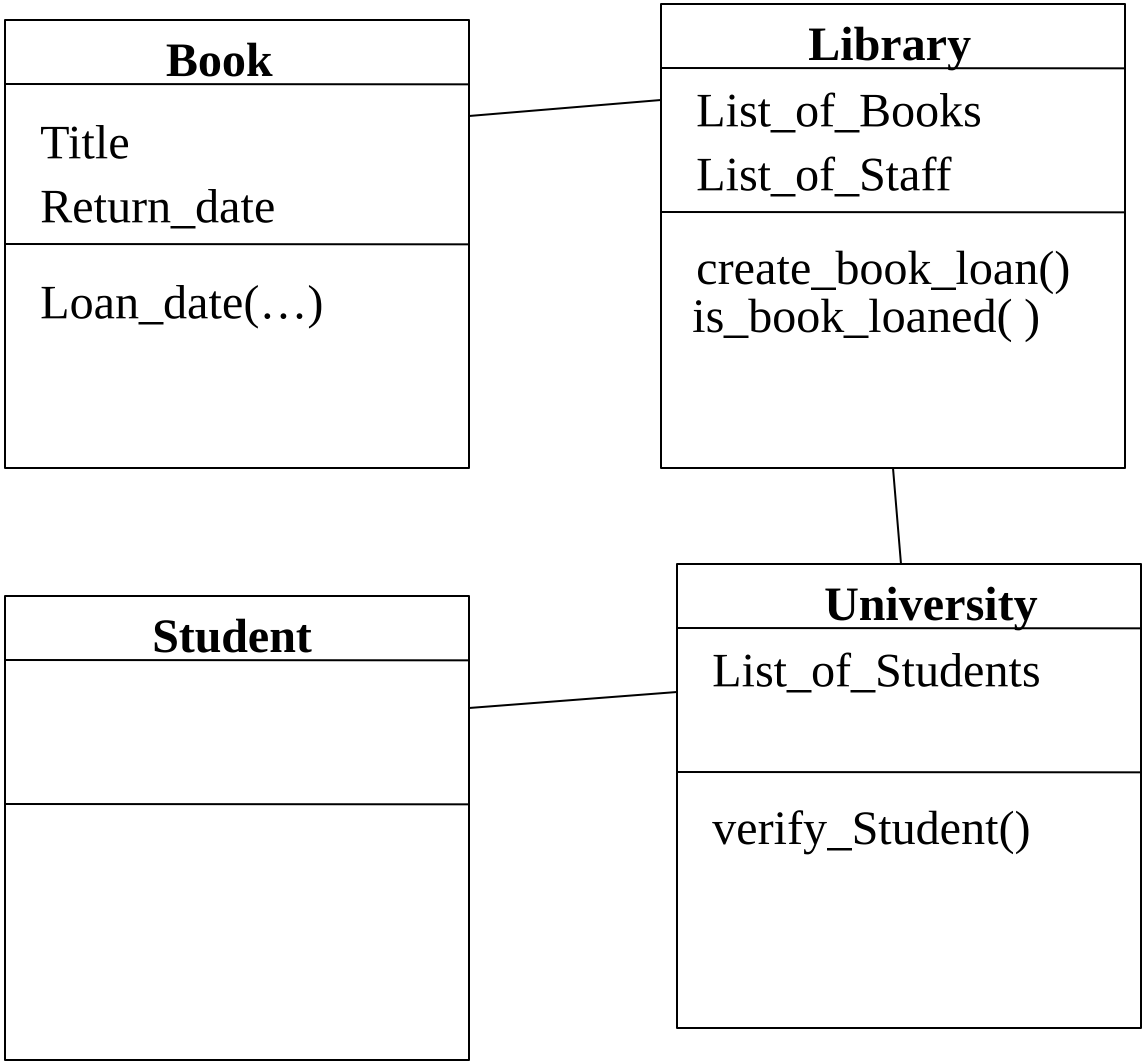
- **Problem Statement:**

  The Eng & Comp Sci library is part of the University of Toronto libraries. You were contacted by the library to develop a software for managing book loans. The library has 15,000 titles serving 40,000 students. An undergraduate student is allowed to borrow a book for up to 2 weeks and can borrow up to 5 books at a time. A graduate student is allowed to borrow a book for up to 4 weeks and can borrow up to 10 books. The library has a staff of 4 employees (librarians). An employee would be interested to know if a book is borrowed or in the library premise, if a book is a borrowed, how is borrowing it and when will it return.

LP: Library Software Case Study

- **Software Design (for an Imperative Language):**





Book

Title
Return_date

Loan_date(…)

Library

List_of_Books
List_of_Staff

create_book_loan()
is_book_loaned( )

Student

University

List_of_Students

verify_Student()

LP: Library Software Case Study

- **Software Design (for a logical language):**

- The library has 15,000 titles → fact
- The library serves 40,000 students → fact
- The library has a staff of 4 librarians → fact

- An undergraduate student is allowed to
borrow a book for up to 2 weeks → business rule
- An undergraduate can borrow up to
5 books at a time → business rule
- A graduate student is allowed to
borrow a book for up to 4 weeks → business rule
- A graduate can borrow up to 10 books → business rule
-* A student must be a UoT student → business rule
-* Only one student can borrow a book → business rule

- A librarian would be interested to know
if a book is borrowed or in the library premise→ functionality/query
- A librarian would be interested to know
who is borrowing a book → functionality/query
- A librarian would be interested to know
when will a book return. → functionality/query

LP: introduction – cont’d

- **Predicate calculus allows us to represent facts, business rules, and queries as logical statements.**

- **By transforming a problem statement to a predicate calculus, we can try to proof a statement!, use induction and deduction, etc…**

LP: Library Software Case Study

- **Software Design (for a logical language):**

- The library has 15,000 titles → fact → true

- The library serves 40,000 students → fact → true

- The library has a staff of 4 librarians → fact → true

- An undergraduate student is allowed to borrow a book for up to 2 weeks → business rule

- An undergraduate can borrow up to 5 books at a time → business rule

- A graduate student is allowed to borrow a book for up to 4 weeks → business rule

- A graduate can borrow up to 10 books → business rule

-* A student must be a UoT student → business rule

-* Only one student can borrow a book → business rule

if (cond)
  true
else
  false

- A librarian would be interested to know if a book is borrowed or in the library premise → functionality/query

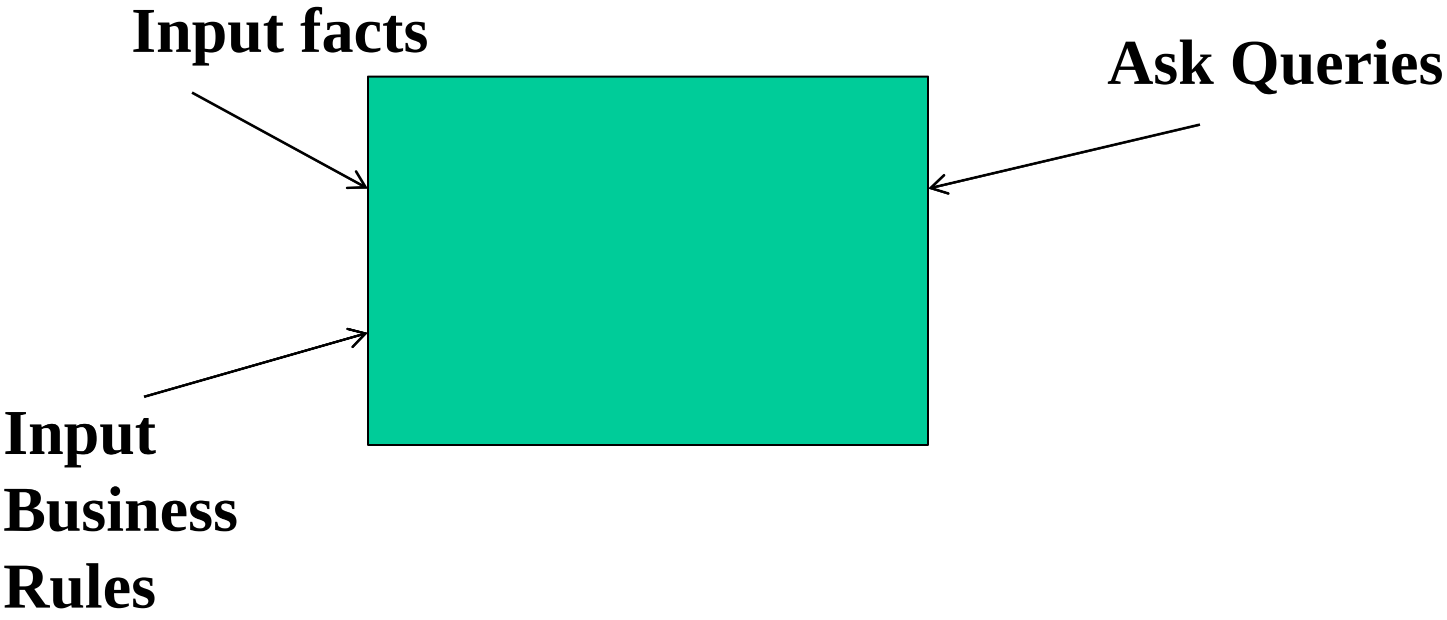
- A librarian would be interested to know who is borrowing a book → functionality/query

- A librarian would be interested to know when will a book return. → functionality/query

Find value of X such that *f*(X) = true

LP: introduction – cont’d

- In logical programming, the model is



Input facts

Ask Queries

Input
Business
Rules

- How can we translate a problem statement to predicate calculus/logic?

LP: operators in predicate calculus

- **Connectors:**

Name	Book	Ex	Alt	Meaning
negation	¬	¬a	! ~	not a
conjunction	∩	a∩b	∧ & ,	a and b
disjunction	∪	a∪b	∨ |	a or b
equivalence	≡	a ≡ b	=	a equiv to b
implication	⊃	a ⊃ b	⇒	a implies b
		a b	⇐ :-	b implies a

Precedence: ¬ then ∩ ∪ ≡ then ⊃

Examples:

$a \cap b \supset c$ $\qquad$ $a \wedge b \Rightarrow c$

$a \cup (b \cap c) \supset d$ $\qquad$ $a \vee (b \wedge c) \Rightarrow d$

- **Quantifiers:**

Universal	$\forall X.P$	For all X, P is true
Existential	$\exists X.P$	There exists a value of X such that P is true

(often leave out the ".")

Examples:

$\forall X.(\text{teachingFaculty}(X) \Rightarrow \text{facultyMember}(X))$

$\forall X.(\text{teachingFaculty}(X) \Rightarrow \exists Y.\text{teaches}(X,Y))$

LP: propositions in predicate calculus

- **A proposition is a *logical statement* that may or may not be true.**

- **Consists of *objects* and their *relationships* to each other**

- **Propositions are written in a *mathematical function form***
  - E.g. A is a B or A is B == written as B(A)

- **Propositions have no intrinsic semantics.**
  - Do not supply meaning, just ids. We are actually interpreting them.

LP: atomic vs. compound propositions

- **Atomic Proposition:**
  - Simplest form of logical statements
  - Made up of two parts: *functor* and *parameters*
  - E.g.

Mary is a woman	woman(mary)
Tom and Mary are married	married(tom,mary)
Scott teaches CSC341 in Summer	teaches(scott, CSC341, Summer)

- **Compound propositions:**
  - Two or more atomic propositions connected with *logical connectors*
  - E.g.

Tom is either smart or dumb	smart(tom) V dumb(tom)
Tom is not dumb	¬dumb(tom)
Tom is married to someone	(∃X) [married(tom,X)]
Tom loves everything	(∀X) [loves(tom,X)]
Tom is married to a human female	

(∃X) [married(tom,X) Λ female(X) Λ human(X)]

LP: implication in predicate calculus

- **Propositions related with each other by an *if-then* semantics, can be expressed using logical implication (denoted by ➔)**

- **Examples:**
  - If someone breaks the law, then she/he will be sent to jail or given a fine but not both.
    - P is breaks the law , Q is sent to jail , R is given a fine, v is a variable
    - $P(v) \rightarrow [(Q(v) \lor R(v)) \land \neg(Q(v) \land R(v))]$
  - If December is a cold dark month then January is a cold dark month
    - P is dark, Q is cold , R is a month, d December and j January
    - $[P(d) \land Q(d) \land R(d)] \rightarrow [P(j) \land Q(j) \land R(j)]$
    - Literally: *if December is cold, and December is dark and December is a month then January is cold, and January is dark and January is a month*
  - There exists at least one x, such that x is a country and x is ruled by a Queen.
    - P is a country , Q ruled by a Queen , x is a variable
    - $(\exists x)\ (P(x) \land Q(x))$
    - Literally: *there is an X that is both P and Q*

LP: implication in predicate calculus

- **Examples cont’d:**
  - Every person who is smart is also rich:
    - $(\forall X)\ [\text{person}(X) \land \text{smart}(X) \Rightarrow \text{rich}(X)]$
  - John has exactly one mother:
    - $(\exists X)\ [\text{mother}(\text{John}, X) \land \text{mother}(\text{John}, Y) \Rightarrow Y = X]$
  - All artists, except poor ones, are rich:
    - $(\forall X)\ [\ (\text{artist}(X) \land \neg\text{poor}(X)\ ) \Rightarrow \text{rich}(X)\ ]$

LP: resolution in predicate calculus

- **We would like to infer new propositions (e.g. facts) from some existing set of propositions.**

- **An inference rule that can be applied atomically is called a *resolution***
  - E.g.

    Given: P1 ← P2 , Q1 ← Q2
    P1 ≡ Q2
    Alternatively: T ← P2 , Q1 ← T
    New rule: Q1 ← P2
    New Set of Rules: P1 ← P2 , Q1 ← Q2 , Q1 ← P2

- **Resolution gets more complex if variables/values are involved:**
  - To use resolution with variables, we will need to find values for variables that allow matching to proceed.
  - E.g.

    Given: F(X,Y) ← P2(Y,X)
    Q1(foo) ← F(foo, bar)
    Is this a New rule? Q1(foo) ← P2(bar, foo)