Logic Programming

Department of Computer Science
California State University, Sacramento
(adopted from Prof. Du Zhang's Notes)

Main Concepts Covered

- Horn clause logic programs.
- First order predicate calculus:
  - Language.
  - Logical equivalence.
  - Models, validity and inconsistency of wff.
  - Logical consequence.
  - Soundness and completeness of inference rules.
- Resolution principle:
  - Unification.
  - Most general unifier.
  - Resolution refutation.
- Handling of negation:
  - Closed-world assumption.
  - Negation as failure.

Main Features of LP

- LP is a programming paradigm in which
  - Logic is used to represent programs (Horn clauses).
  - Deductions are used as computations (resolution).
- The strength of LP paradigm:
  - Describe what the logical structure of a problem solution is (rather than making us prescribe how computer is to go about solving the problem).
  - Instead of human learning to think in terms of computer operations, the computer should perform instructions that are easy for human to specify.

Benefits

- Well-defined semantics.
- Separation of logic from control:
  - Algorithm = Logic (what) + Control (how).
  - Ideal: only need to specify the logic component, leave the control totally to system.
- Opportunities for various parallelisms.
- Partial computation, and invertability.

Alphabets

- Alphabet of symbols
  - Variables.
  - Constants.
  - Functions.
  - Predicates.
  - Connectives: $\rightarrow$, $\neg$, $\wedge$, $\vee$, $\leftrightarrow$.
  - Quantifiers: $\forall$, $\exists$.
  - Punctuations: parentheses, comma, period.
- Some notes:
  - Last three sets are the same for every alphabet.
  - First four sets vary from alphabet to alphabet.
  - Only second and third sets may be empty.

Logical Equivalence

- If truth values of two wff are the same under every interpretation, then they are logically equivalent.
- Set of equivalences:

$G \rightarrow H \equiv \neg G \vee H$

$\neg(G \vee H) \equiv \neg G \wedge \neg H$

$\neg(\exists x\, G(x)) \equiv \forall x\, (\neg G(x))$

……

Models

- An interpretation *I* is said to be a model for a wff *G* if *G* is evaluated to true in *I*.
- An interpretation *I* is said to be a model for a set P of wff if every wff in P is evaluated to true in *I*.

Validity and Inconsistency

- A wff G is consistent iff there exists a model for G.
- A wff G is inconsistent iff there exists no model for G.
- A wff G is valid iff G is true in all interpretations.
- A wff G is invalid iff there exists at least one interpretation under which G is false.
- Some relations:
  - A wff is valid iff its negation is inconsistent.
  - A wff is inconsistent iff its negation is valid.
  - If a wff is valid, then it's consistent, but not vice versa.
  - If a wff is inconsistent, then it's invalid, but not vice versa.

Logical Consequence

- Decide whether one statement follows from some other statements.
- Given a set of wff {H1,…,Hn}, and a wff G, G is said to be a logical consequence of {H1,…,Hn} iff every model of {H1,…,Hn} is also a model of G.

Clauses

- Normal forms:
  - Conjunctive NF.
  - Disjunctive NF.
  - Prenex NF.
  - Skolem NF.
- A clause is a wff consisting of disjunction of literals.
- Any wff can be converted to a set of clauses (using logical equivalences).

Horn Clauses

- Horn clauses are clauses that have at most one positive literal:
  - Facts (assertions).
  - Rules (H:-C1,…,Cn).
  - Queries (?-G1,…,Gn).

  - Note:
    - B :- A means A → B or ~A v B
    - It can also be read as " if A is true then B is true"
    - or "B is true if A is true"

Example

- Consider the following sentence : 'All men are mortal' We can express this thing in Prolog by :

  mortal(X) :- human(X).
- The clause can be read as 'X is mortal if X is human'.

Horn Clauses

- Drop the quantifiers (i.e., assume them implicitly). Distinguish variables from constants, predicates, and functions by upper/lower case:
- parent(X,Y) $\rightarrow$ ancestor(X,Y).
  ancestor(A,B) and ancestor(B,C) $\rightarrow$ ancestor(A,C).
  mother(X,Y) $\rightarrow$ parent(X,Y).
  father(X,Y) $\rightarrow$ parent(X,Y).
  father(bill,jill).
  mother(jill,sam).
  father(bob,sam).

(Horn Clause) Logic Programs

- Representing general knowledge about objects and their relationships (in terms of rules).
- Asserting instances of relationships for objects (in terms of facts).
- Asking questions about objects in some relationships.

An Example

Map coloring problem.

```
mapColor(X1,X2,X3,X4,X5):-next(X1,X2),next(X1,X3),next(X1,X4),
                          next(X2,X3),next(X2,X5),next(X3,X4),
                          next(X3,X5), next(X4,X5).

next(blue, green).
next(blue, red).
next(blue, yellow).
next(green, blue).
next(green, red).
next(green, yellow).
next(red, green).
next(red, blue).
next(red, yellow).
next(yellow, green).
next(yellow, red).
next(yellow, blue).
```

```
?- mapColor(red, X,Y,Z,U).

X= green
Y= blue
Z= green
U= red
……
```

Facts

- Facts are expressed in terms of a particular relation holding among objects.
- Names of relations and objects
  - Begin with a lower-case letter.
  - Arbitrary.
- Template for a fact
  - *rel_name(obj1, ..., objn)*
  - Predicate as *rel_name*
  - Arity of predicate
  - Order of objects.
- Interpretation of objects and relations.
- Potential discrepancy between a fact in program and a fact in real world.
- A database: collection of facts.

Questions (queries)

- Are expressed in terms of a relation among objects (but with a different interpretation).
- ?- *rel_name*(*obj1, ..., objn*)
- Search for answers to a question (top-down process).
  - Two facts match if their predicates are the same, and
  - Their corresponding arguments each are the same.
- Prolog returns a "yes" if a match is found; otherwise, "no"
  - "no" means nothing in the database matches the question (or the question is not provable by the facts in database).
  - "no" does not necessarily mean that the fact asked in the question is false (database may be incomplete).

Variables

- Used to stand for objects in a relation which we do not know and would like Prolog to find out.
- Any name beginning with a capital letter

  ?-likes(john, X)
- Instantiation of variables
  - If there exists an object that a variable stands for, then the variable can be instantiated.
  - Such instantiations will be used as the answers to a W-question.
- Search for answers
  - Same process (top-down search).
  - Processing uninstantiated variables.
  - One solution vs. all solutions.

An Example and In-class EX

Who teaches what.

f1: teaches(john, database, s1).
f2: teaches(john, ai, s1).
f3: teaches(mary, os, s1).
f4: teaches(mary, ai, s2).
f5: teaches(paul, compiler, s1).

Query 1: What's the course taught by John and Mary?

```
?- teaches(john, X, _), teaches(mary, X, _).
```

Query 2: Who teaches both os and compiler?

```
?- teaches(X, os, _), teaches(X, compiler, _).
```

Anonymous Variables

- Used to ignore values you are not interested in.
- Represented as a single underscore "_", can be used in place of any variable, matches anything, and will never get set to a value.
- Several anonymous variables in the same relation need not be given consistent interpretation.

Conjunctions

- Each question contains two separate sub-questions Prolog must satisfy.
- "and" in each question indicates that we are interested in the conjunction of two sub-questions.
- Conjunction is represented as a "comma" (,).

Search Process

- Subgoal satisfaction: left-to-right.
- Use of facts: top-down.
- Search process for Q2.
- Try Q1, but with subgoals reversed.
- Place-marker for each subgoal.
- Success condition: ?
- Failure condition: ?

Backtracking

- Any time a subgoal fails, Prolog goes back and tries to resatisfy its left neighbor. The behavior is called "backtracking".
- If a variable becomes instantiated, all occurrences of the same variable in the question become instantiated.
- If a variable becomes uninstantiated due to backtracking, then all occurrences of it in the question also become uninstantiated.
- One solution vs. all solutions.

Parallelisms

- AND parallelism:
  - Ability to pursue several subgoals in parallel.
  - ?- p(…, X,…), q(…, X,…), r(…,X,…).
- OR parallelism:
  - Ability to pursue several alternative solutions (in database) in parallel.

Rules

- General knowledge about objects and their relationships.
- A rule consists of:
  - Head: that defines a relation.
  - Implication sign :-
  - Body: conjunctive conditions that define the head.
- Scope of variables.

A Logic Program

- Consists of:
  - A set of rules.
  - A set of facts.
- In the pure declarative programming, the following should be immaterial:
  - Order of conjunctive conditions in a rule.
  - Order of the rules.
  - Order of facts.
- Those orders would have impact in a Prolog implementation.

An Example

```
1:  ancester(X,Y) :- parent(X,Y).
2:  ancester(X,Y) :- parent(X,Z), ancester(Z,Y).
3:  parent(dave, mike).
4:  parent(mike, john).

?- ancester(X, john).      ; how to obtain results
                           ; search space.
                           ; invertability, input/outputs nondet.
                           ; e.g., ?- ancester(dave, X).
```

The Cut

- Control mechanisms:
  - Computation (left-to-right).
  - Search (top-to-bottom).
  - Extra logical features.
- The cut: reduce search space by pruning the search tree (foo:- a, b, c, !, d, e, f.).
- syntactical aspects:
  - Special built-in predicate "!" with no argument.
  - As a goal, it succeeds immediately.
- semantic aspects:
  - Cannot be resatisfied, thus, committing the system to all the decisions made before the cut
  - All other alternatives are discarded when backtracking.

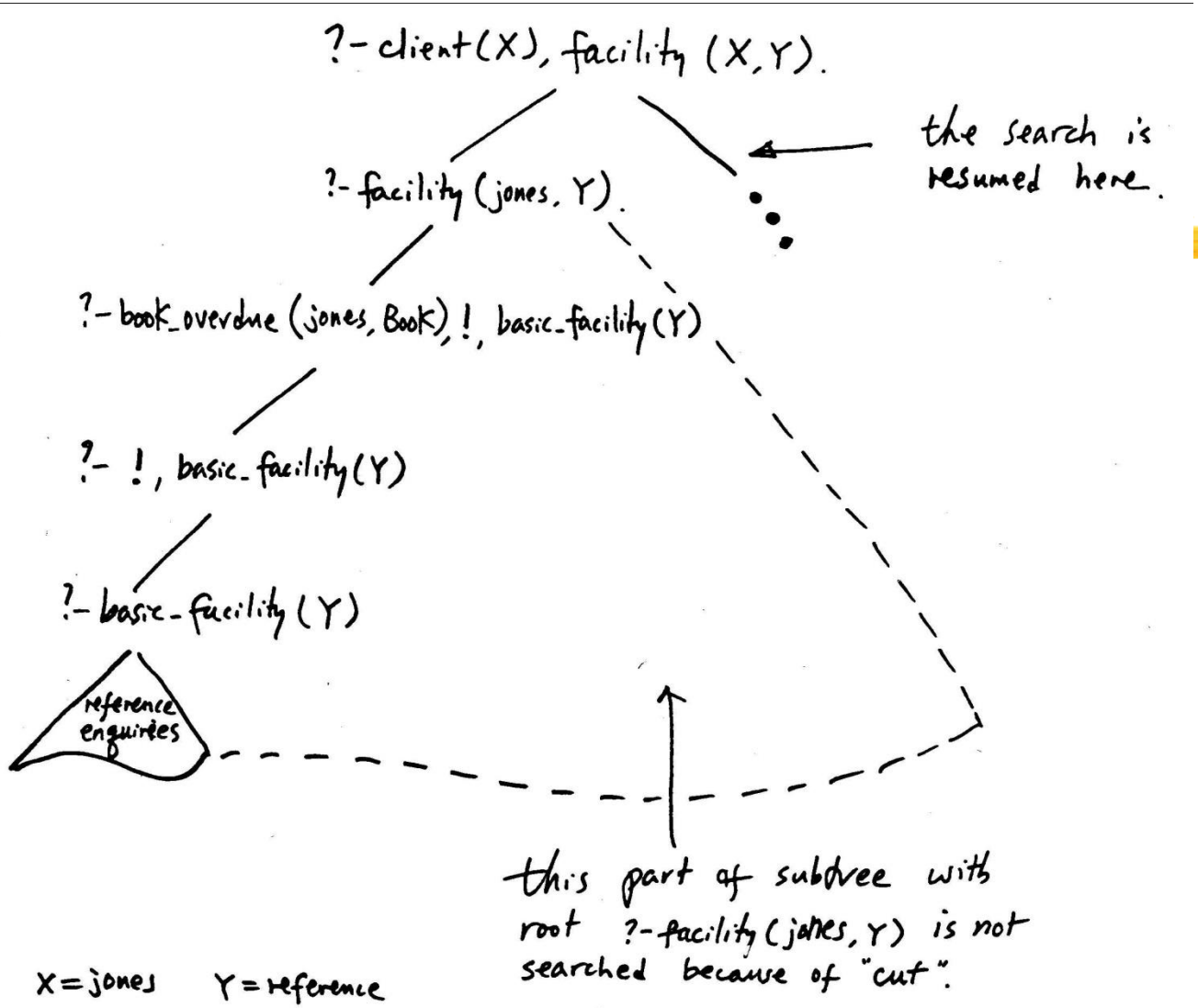
Common Uses of Cut

- Tell Prolog that the right rule has been found.
- Tell Prolog to terminate the generation of alternative solutions through backtracking.

Examples

```
1.  facility(P,F):-bookOverdue(P,B), !, basicFacility(F).
2.  facility(P,F):-generalFacility(F).        /* order here is important */
3.  basicFacility(reference).
4.  basicFacility(inquiry).
5.  additionalFacility(borrow).
6.  additionalFacility(interLibLoan).
7.  generalFacility(X):-basicFacility(X).
8.  generalFacility(X):-additionalFacility(X).
9.  bookOverdue(doe, book1).
10. bookOverdue(doe, book3).
11. bookOverdue(jones, book2).
12. client(doe).
13. client(jones).
14. client(smith).

    ?- client(X), facility(X, Y).
```


?- client(X), facility(X,Y).
the search is resumed here.
?- facility(jones, Y).
?- book_overdue(jones, Book), !, basic_facility(Y)
?- !, basic_facility(Y)
?- basic_facility(Y)
reference enquiries
this part of subtree with root ?- facility(jones, Y) is not searched because of "cut".
X = jones Y = reference
X = jones Y = enquiries
X = metesk Y = reference
X = metesk Y = enquiries
X = metesk Y = borrowing
X = metesk Y = inter_library_loan

Notes for Homework

- Download SWI-Prolog to your computer
- Edit your source code of Prolog and save it into local space (C drive)
- Pay attention to syntax requirements (case sensitive and end with “,” or “;”)
- type “.” for one answer, and then type “;” for more answers