Logic Programming

Proposals Based on Logic Languages

- SDN/Control-plane languages
  - Examples: Flog, Flowlog, NDlog, etc.
- Distributed systems / overlay networks
  - Examples: P2, Overlog, R/Overlog, BOOM, etc.

Prolog

- General purpose logic based language
  - Probably the most popular logic based language, but there are many others (e.g., miniKanren)
- Developed in 1972 by Alain Colmerauer and Philippe Roussel
- Based on first-order logic, and used for AI or linguistics.

How to write and run programs

```
$ /usr/local/bin/swipl  /* start the interpreter */

?- help(help).          /* get help */

?- [graph].             /* load the file named graph.pl */
```

Facts

```
teacher(robert,pl).
student(marco,pl).
student(marco,os).
```

- A fact starts with a predicate, ends with a period.
- The predicate may be follows by one or more arguments in parenthesis.
- A fact indicates that a statement is true.

Variables

```
teacher(robert,X).
Yes. X = pl.
```

- Capitalized terms indicate a logical variable.
- You can use a variable to ask a query.
- Queries are answered through *unification*.

Quantification

- Notice that in *queries* X is *existentially quantified.*
  - *“teacher(pezze, plclass).” is does there exist an X, such that the relation teacher(X,plclass) holds?*
- In *facts*, the variable is implicitly *universally quantified.*
  - *“teacher(X, plclass).” is for all X, there is a relation teacher(X,plclass).*

Unification

- A process of solving equations between symbolic expressions
- Part of the theory of *resolution*
- The idealized programming model relies on angelic (i.e., always correct) non-determinism
- Many applications, including type inference and logic programming

Rules

```
father(robert,emilie).
father(robert,julie).
mother(susie,emilie).
mother(susie,julie).
parent(X, Y) :- father(X, Y).
parent(X, Y) :- mother(X, Y).
grandfather(X,Y) :- father(X,Z), parent(Z,Y).
```

- the rule head is true, if the rule body is true
- , means conjunction. ; means disjunction.
- Rules are written as Horn clauses

Implication Elimination

- grandparent(X,Y) ← parent(X,Z), parent(Y,Z).
- This is a form of the law of *universal modus ponens.* From the rule R : A ← B_1,…B_n, and the facts B’_1,…B’_n, then A’ can be deduced if A’ ← B’_1,…B’_n is an instance of R.

Lists

```
append([X|Xs],Ys,[X|Z]) :- append(Xs,Ys,Z).
append([],Ys,Ys).

append([a,b],[c,d],Z)?
yes, Z = [a,b,c,d]
```

- As a syntactic convenience, cons(X,Y) is typically written as [X | Y].
- cons(a,cons(b,nil)) is written [a,b,c], and [] denotes the empty list.

Another List Examples

```
reverse([],[]).
reverse([X|Xs],[Y|X]) :- reverse(Xs, Y).
```

Logic Program Interpreter

Input: A logic program P and a goal G
Output: G θ, the instance of G deduced P, or failure.
Algorithm:
    Initialize the resolvent to be G, the input goal.
    While the resolvent is not empty do
        Choose a goal A from the resolvent and a (renamed) clause
        A' ← B_1,B_2, ..., B_n
        from P such that A and A' unify (exit if no such
        goal and clause exist).
        Remove A from the resolvent and add B1,B2, ..., Bn to the
        resolvent.
        Apply θ to the resolvent and to G.
end while
If the resolvent is empty then output G, else outputfailure.

Predicates

```
maxcomp(X,Y,X) :- X >= Y.
maxcomp(X,Y,Y) :- X < Y.
max([],0).
max([X|Xs], X) :- max(Xs, Y), maxcomp(X, Y, X).
max([X|Xs], Y) :- max(Xs, Y), maxcomp(X, Y, Y).
```

- Several build in predicates: <, >, =<, >=, is, =, =\=
- Note: 4 = 1 + 3. no. 4 is 1+3. yes.

Factorial Example

```
factorial(0,1).

factorial(A,B) :-
            A > 0,
            C is A-1,
            factorial(C,D),
            B is A*D.
```

Cut

```
merge([X|Xs],[Y|Ys],[X|Zs]) :- X<Y, !,
  merge([Xs], [Y|Ys],Zs).
```

- A predicate called cut, !, is used to prune the search tree generated by backtracking.
- Operational definition: the goal succeeds and commits Prolog to all the choices made since the parent goal was unified with the head of the clause the cut occurs in.

Cut for Negation

```
not X :- X,!, fail.
not X.
```

- You can force failure using the cut and fail predicates
- Provides a way to implement negation

Datalog

Datalog

- Syntax is a subset of prolog
- Ofter used in deductive databases (or as an alternative to SQL)
- Evaluation is “bottom up”, not “top down”.

Datalog Evaluation

Input: Datalog program P and input instance $\mathbf{I}$

Output: $P(\mathbf{I})$

1. Set P' to be the rules in P with no *idb* predicate in the body;
2. $S^0 := \emptyset$, for each *idb* predicate S;
3. $\Delta_S^1 := P'(\mathbf{I})(S)$, for each *idb* predicate S;
4. $i := 1$;
5. do begin
   for each *idb* predicate S, where $T_1, \ldots, T_l$
   are the *idb* predicates involved in rules defining S,
   begin
   $S^i := S^{i-1} \cup \Delta_S^i$;
   $\Delta_S^{i+1} := P_S^i(\mathbf{I}, T_1^{i-1}, \ldots, T_l^{i-1}, T_1^i, \ldots, T_l^i, \Delta_{T_1}^i, \ldots, \Delta_{T_l}^i) - S^i$;
   end;
   $i := i + 1$
   end
   until $\Delta_S^i = \emptyset$ for each *idb* predicate S.
6. $s := s^i$, for each *idb* predicate S. ■