Note 3

CS 475

February 7, 2003

1 Important Definitions/Notations

Datalog (syntax, semantics)
Proof procedures (bottom-up, top-down): ground vs. non-grounded (with variables)
Resolution: grounded vs. with variables
Substitution
Unifier (mgu)

Remark: on selection of a clause in the proof procedures. In both procedures, we have to select a clause that satisfies certain conditions. Often, there are several possibilities. The selection is called a *non-deterministically selection.*

Two possible treatment:

- *Don't-care nondeterminism*: If one selection doesn't lead to a solution, there is no point trying other alternatives. (in bottom-up procedure) When the set of possible selections is infinite, the selection policy must ensure *fainess*. That is, a possible section will eventually be selected.
- *Don't-know nondeterminism*: If one choice doesn't lead to a solution, other choices may. (in top-down procedure) All possible choices have to be examined in order to answer 'NO'.

2 Function Symbols

The Datalog language is very simple. Too simple indeed. It forces us to have name for every individual. It does not allow us to describe an object in term of its component. Often we want to refer to individuals in terms of components.

Examples: 4:55 p.m (two parts: the time, the indicator). English sentences (possible parts: noun, verb, adjective, etc.). A classlist (possible parts: student names, ids, class title, class code, etc.)

To allow objects to be described in term of components, we add to the language a set of function symbols, each is assocaited with a number of parameters. We extend the notion of term. So that a term can be $f(t_1, \ldots, t_n)$ where f is a function symbol and the t_i are terms.

We also need to extend the definition of an interpretation by adding to it a mapping that maps a n-ary function symbol into a function from D^n to D. In an interpretation and with a variable assignment, term $f(t_1, \ldots, t_n)$ denotes an individual in the domain.

2.1 The Power of Function Symbols

Without function symbol, we will need infinitely many number of constants for the representation of a domain with infinitely many individuals. With one function symbol and one constant we can refer to infinitely many individuals.

Example: The KB with two clauses:

$$number(s(X)) \leftarrow number(X)$$
$$number(0).$$

describes the set of integer numbers.

Function symbols also allow us to write structural recursive programs. Useful data structures as terms: a binary tree can be represented by a set of nodes whose components are name, the left tree, and the right tree (using: $node(N, LT, RT)$) and a set of leaf nodes ($leaf(L)$). We can define the predicate $at\_leaf(L, T)$ that is true when L is the label of a leaf in tree T:

$$at\_leaf(L, leaf(L)) \leftarrow$$
$$at\_leaf(L, node(N, LT, RT)) \leftarrow at\_leaf(L, LT).$$
$$at\_leaf(L, node(N, LT, RT)) \leftarrow at\_leaf(L, RT).$$

Similarly, the relation $in\_tree(N, T)$ is true if N is an interior node of T. This can be defined as follows:

$$in\_tree(N, node(N, LT, RT)) \leftarrow$$
$$in\_tree(L, node(N, LT, RT)) \leftarrow in\_tree(L, LT).$$
$$in\_tree(L, node(N, LT, RT)) \leftarrow in\_tree(L, RT).$$

An important function symbol in Datalog – Lists: A list is an ordered sequence of elements. Let's use the constant *nil* to denote the empty list, and the function $cons(H, T)$ to denote the list with first element H and rest-of-list T. These are not built-in. The list containing *david*, *alan* and *randy* is

$$cons(david, cons(alan, cons(randy, nil)))$$

$append(X, Y, Z)$ is true if list Z contains the elements of X followed by the elements of Y

$$append(nil, Z, Z).$$
$$append(cons(A, X), Y, cons(A, Z)) \leftarrow append(X, Y, Z).$$

2.2 Proof Procedures with Function Symbols

Proof procedures with variables can be used. Imporant in selection the clause: *fairness* for bottom-up procedure and *no-unification between X and a term containing X* for top-down procedure.

Example: Without fairness, trying to proof $number(0)$ will fail.

$$number(s(X)) \leftarrow number(X)$$
$$number(0).$$

Example: The KB $lt(X, s(X))$ would not entail $lt(X, X)$.

Example: Reasoning with function symbols and variables .PPT file

Example: Derivation of the queries in Exercise 2.7

Example: Exercise 2.11