CS345 Notes for Lecture 11/25/96

NAIL

NAIL (Not Another Implementation of Logic) was a prototype deductive database system developed at Stanford.

- The principal designers and implementors were Marcia Derr, Shinichi Morishita, Kate Morris, and Geoff Phipps.
  - □ The latter two were wed last year.

Structure of NAIL

1. GLUE: logical rules plus conventional flow-of-control, procedures, etc.
2. NAIL: conventional logic front end. Translated via magic-sets and well-founded negation to GLUE procedures.

GLUE Assignments

Assignment statements look like rules, but *they are executed whenever, and only when, they are reached during program execution.*

- Four types of assignment: Clear-and-assign (`:=`), insert (`:+`), delete (`:-`), update (`:=[Z]`).

Example:

```
p(X,Y) := q(X,Z) & r(Z,Y)
```

- Means $p := \pi_{1,3}(q \bowtie r)$.
- Executed as written whenever control flow reaches this statement.

Three Types of Subgoals

GLUE rules can have three types of subgoals.

1. EDB: stored physically in a subdirectory called `edb`.
2. IDB: defined by NAIL rules included with the GLUE program.

3. Procedural. These subgoals have their value defined by a GLUE procedure. GLUE procedures have input and output arguments, and the input arguments must be bound when this subgoal is reached in a left-to-right evaluation of the GLUE statement.

Control Flow Statements

While-statement:

while <condition> **do** <statements> **end**;

- Condition is a rule body.
- Its value is "true" iff the body is nonempty.
- Similar idea: repeat, loop, if-then-else statements.

Example:

```
module semi;
    export main(:);
    edb arc(X,Y), path(X,Y);
    prog main(:)
        rels newPath(X,Y);
        arc(X,Y) := {}.
        arc(1,2).
        arc(2,3).
        arc(3,4).
        arc(4,5).
        arc(2,4).
        newPath(X,Y) := arc(X,Y).
        path(X,Y) := {}.
        while newPath(X,Y) do
            path(X,Y) += newPath(X,Y).
            newPath(X,Y) := newPath(X,Z) & arc(Z,Y).
            newPath(X,Y) -= path(X,Y).
        end;
    end
end
```

This program performs seminaive evaluation for the rules

```
path(X,Y) :- arc(X,Y)
path(X,Y) :- path(X,Z) & arc(Z,Y)
```

- It is not necessary to write this program in GLUE; the NAIL compiler produces it automatically from the rules.
- Line (12): On round 1, the new paths are the arcs.
- Line (13): Initially, *path* is empty.
- Line (14): Go around the loop until there are no new paths discovered on some round.
- Line (15): Copy the new paths into the full set of paths.
- Line (16): Compute "new new" paths by extending only the new paths by one more arc.
- Line (17): Throw away from the new new paths those paths that are actually old.

HiLog

- A value in some relation can be used as (part of) the name of another relation.
- Relation names can be nontrivial terms.

Example: Suppose every drug company C has a relation whose name is $data(C)(Drug, Dosage)$.

- $data(C)$ is the name of the relation, e.g., $data(serle)$, $data(merck)$.
- Each relation of this type has two attributes, a drug and its dosage.

A master relation $drugs(Drug, Manf)$ tells us where to look for dosage information.

- This actually makes sense if the $data(C)$ relations are distributed and NAIL is used as an integrating query language.
- To get dosage information for a given drug, we could write in NAIL:

```
dosage(Drug,Dos) :-
        drugs(Drug,Manf) &
        data(Manf)(Drug,Dos).
```

- NAIL allows variables in predicate names, as long as the *principal functor* (outermost symbol of a term) is bound when the subgoal is reached.

Example: Not allowed:

```
value(X) := R(X).
```

"Find all values that appear in some unary relation."

Example: Allowed:

```
dosage(Drug,Dos) :-
        drugs(Drug,Manf) &
        Manf(Drug,Dos).
```

Provided the name of the "data" relation for each company C is just C itself, e.g., $serle(Drug, Dos)$.

References

1. Derr, Morishita, and Phipps, "Design and implementation of the Glue-Nail database system," 1993 SIGMOD, pp. 147–156.
2. Ramakrishnan, Srivastava, Sudarshan, and Seshadri, "Implementation of the CORAL database system," *ibid.*, pp. 167–176.
3. Ramakrishnan, Roth, Seshadri, Srivastava, and Sudarshan, "The CORAL deductive database system," *ibid.*, pp. 544–545.
4. Chen, Kifer, and Warren, "HiLog: a first order semantics for higher order programming constructs," in *Second Intl. Workshop on Database Programming Languages*, Morgan-Kaufmann, San Mateo, 1989.