Five Groups of Rules for Magic Construction

Let r be a typical rule

$$H \text{ :- } G_1 \text{ \& } G_2 \text{ \& } \cdots \text{ \& } G_n$$

Group I

Supplementary $\Rightarrow$ magic for next subgoal. If G_i has IDB predicate p:

$$m\_p(\text{bound args of } G_i) \text{ :- } sup_{r.i-1}(\text{variables})$$

Group II

Magic $\Rightarrow$ 0th supplementary. If head has predicate q:

$$sup_{r.0}(\text{variables}) \text{ :- } m\_q(\text{bound args})$$

Group III

$i-1$st supplementary + $G_i \Rightarrow i$th supplementary.

$$sup_{r.i}(\text{variables}) \text{ :- } sup_{r.i-1}(\text{variables}) \text{ \& } G_i$$

Group IV

Last supplementary + last subgoal $\Rightarrow$ head.

$$H \text{ :- } sup_{r.n-1}(\text{variables}) \text{ \& } G_n$$

Group V

Initialize. If query has predicate p we have the bodyless rule

$$m\_p(\text{bound args from query})$$

- Group V is the only rule that depends on the actual query. Others depend only on the binding *pattern*.

Magic-Sets Beats Top-Down

- Claim: the "magic" rules implemented by seminaive evaluation only infers facts that any top-down implementation would infer.
  - I.e., magic-sets + seminaive has the advantages of both top-down and bottom-up.
  - Well not exactly. Prolog uses a "tail-recursion elimination" technique that sometimes does in $O(n)$ time what takes $O(n^2)$ by magic-sets.
  - The same trick has been used in deductive systems like LDL, Coral; it is described in Ch. 15 of PDKS-II.

Example

Binary trees constructed as terms.

- Leaves constructed by constant $leaf$.
- Interior nodes constructed by function symbol n. $n(T_1, T_2)$ is a tree with left subtree T_1 and right subtree T_2.
- Predicate $sub(T_1, T_2)$ true when T_1 is a subtree of T_2.
- Predicate $eq(T_1, T_2)$ true when T_1 and T_2 are identical trees.

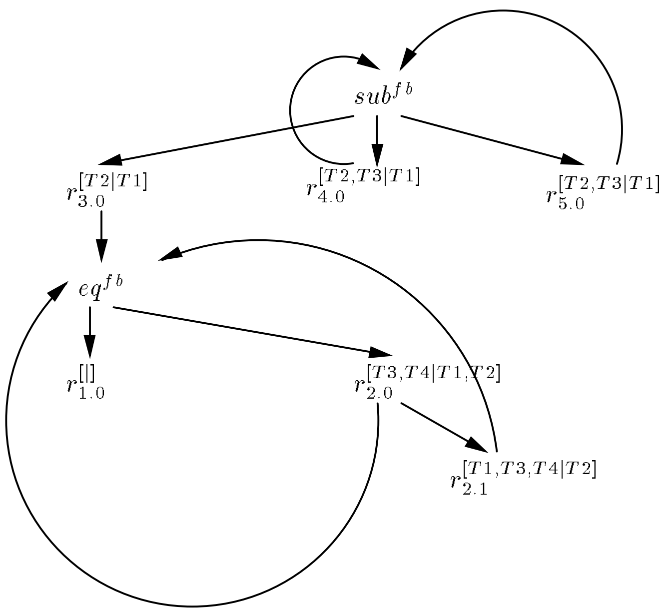
- Rules:

  r_1: `eq(leaf,leaf)`
  r_2: `eq(n(T1,T2),n(T3,T4)) :-`
  `        eq(T1,T3) & eq(T2,T4)`

  r_3: `sub(T1,T2) :- eq(T1,T2)`
  r_4: `sub(T1,n(T2,T3)) :- sub(T1,T2)`
  r_5: `sub(T1,n(T2,T3)) :- sub(T1,T3)`

- Query: `sub(T,n(n(leaf,leaf),leaf))`, i.e., find the subtrees of a specific tree. Adorned goal: sub^{fb}.

The Rule/Goal Graph



- Interesting fact: r_4 and r_5 are not safe. however, for the sub^{fb} binding pattern, all variables of the head either appear in the body *or* are present in a bound argument of the head.

- This notion of *safety with respect to a binding pattern* is appropriate when magic-sets is used.

The Magic Rules

Group I

```
m_eq(T3) :- sup_{2.0}(T3,T4)
m_eq(T4) :- sup_{2.1}(T1,T3,T4)
m_eq(T2) :- sup_{3.0}(T2)
m_sub(T2) :- sup_{4.0}(T2,T3)
m_sub(T3) :- sup_{5.0}(T2,T3)
```

Group II

```
sup_{2.0}(T3,T4) :- m_eq(n(T3,T4))
sup_{3.0}(T2) :- m_sub(T2)
sup_{4.0}(T2,T3) :- m_sub(n(T2,T3))
sup_{5.0}(T2,T3) :- m_sub(n(T2,T3))
```

Group III

```
sup_{2.1}(T1,T3,T4) :-
        sup_{2.0}(T3,T4) & eq(T1,T3)
```

Group IV

```
eq(leaf,leaf)
eq(n(T1,T2),n(T3,T4)) :-
        sup_{2.1}(T1,T3,T4) & eq(T2,T4)
sub(T1,T2) :- sup_{3.0}(T2) & eq(T1,T2)
sub(T1,n(T2,T3)) :-
        sup_{4.0}(T2,T3) & sub(T1,T2)
sub(T1,n(T2,T3)) :-
        sup_{5.0}(T2,T3) & sub(T1,T3)
```

Group V

```
m_sub(n(n(leaf,leaf),leaf))
```

Simplifying Magic Rules

- Use Group II rules to replace $sup_{r.0}$'s by magic predicates.
- If a supplementary predicate comes before an EDB subgoal, it is used only once and may be eliminated.
- However, if it is before an IDB subgoal, it is used twice, once in Group I and once in Group III or IV.

- Thus, we can omit $sup_{r.i}$ if $(i+1)$st subgoal is EDB and $i > 0$.
- We use in place of Group III rules new rules of the form:

  $sup_{r.j}(\cdots)$:-
  $\quad sup_{r.i-1}(\cdots)$ & G_i & $\cdots$ & G_{j-1}

  provided all of $G_{i+1}, \ldots, G_{j-1}$ are EDB. Also, G_j is IDB, and either G_i is IDB or $i = 1$.

- Similarly, Group IV rules are replaced by

  H :- $sup_{r.i-1}(\cdots)$ & G_i & $\cdots$ & G_n

  provided all of $G_{i+1}, \ldots, G_n$ are EDB. Also, G_i is IDB or $i = 1$.
- In the two rules above, use the appropriate magic predicate if $i = 1$.

Example

For the tree rules, we get:

Group I

```
m_eq(T3) :- m_eq(n(T3,T4))
m_eq(T4) :- sup2.1(T1,T3,T4)
m_eq(T2) :- m_sub(T2)
m_sub(T2) :- m_sub(n(T2,T3))
m_sub(T3) :- m_sub(n(T2,T3))
```

Group III

```
sup2.1(T1,T3,T4) :-
        m_eq(n(T3,T4)) & eq(T1,T3)
```

Group IV

```
eq(leaf,leaf)
eq(n(T1,T2),n(T3,T4)) :-
        sup2.1(T1,T3,T4) & eq(T2,T4)
sub(T1,T2) :- m_sub(T2) & eq(T1,T2)
sub(T1,n(T2,T3)) :-
        m_sub(n(T2,T3)) & sub(T1,T2)
sub(T1,n(T2,T3)) :-
        m_sub(n(T2,T3)) & sub(T1,T3)
```

Group V

```
m_sub(n(n(leaf,leaf),leaf))
```