Section 5 Worksheet

Advanced Datalog

Consider the following schema about software dependencies:

SSpec(s, x) – The software (s) and its specification (x)
SDep(s, d) – The software (s) and a dependency (d) to another program it has
OpenSrc(s) – Software that has been made open source

We assume there are self dependencies or cyclic dependencies otherwise. Also, all dependencies are other software found in SSpec.

1. Find the number of direct dependencies for each piece of software.

Ans(s, n) :- SSpec(s, x), n = count() : SDep(s, d).

2. Find all software (and respective specification) that have more software that is directly dependent than it has direct dependencies

Counts(s, n, m) :- SSpec(s, _), n = count() : SDep(s, _), m = count() : SDep(_, s).

Ans(s, x) :- SSpec(s, x), Counts(s, n, m), n > m.

3. Find all software (and respective specification) that is not dependent on open-source software.

NonAns(d) :- SDep(s, d), OpenSrc(s).
NonAns(d) :- SDep(s, d), NonAns(s).

Ans(s, x) :- SSpec(s, x), !NonAns(s).

4. Find all software (and respective specification) that is not a dependency to any open source software.

NonAns(s) :- OpenSrc(d), SDep(s, d).
NonAns(s) :- NonAns(d), SDep(s, d).

Ans(s, x) :- SSpec(s, x), !NonAns(s).

5. Find the maximum dependency depth for each piece of software.

Depth(s,s,0) :- SSpec(s,_).
Depth(s,d,n+1) :- Depth(s,m,n), SDep(m,d).
Ans(s,n) :- SSpec(s,_), n = max m : Depth(s,_,m).

// English of first rule: There is a downward path from 's' to 's' of length 0.
// Second rule: There is a downward path from 's' to 'd' of length n+1
// if there is a downward path from 's' to 'm' of length n
// and there is a direct dependency from 'm' to 'd'.
// Third rule: The maximum dependency depth of 's'

// is the maximum of downward path lengths from 's' to another software.```

Thought exercise: Would question 5 terminate if cycles in our dependency graph existed? What if we were finding the minimum dependency depth?