w10_datalog

Week 10: Datalog

The Addams Family

Task 1. Capture the Addams Family tree using Horn clauses. To test run the query `parent(X,sharron)` to see if *Sharron* is anyone's parent. The only match should be *Debbie*.

Task 2. A sibling is someone who shares a parent with you. Define a rule `sibling(X,Y)` that captures this. Check that Gomez's sibling is Fester.

Task 3. AbcDatalog allows negation in rules. We can specify that two variables are not equal in a rule by saying `X!=Y`. Check whether `sibling(wednesday,wednesday)` is true, and if so fix it.

Task 4. A cousin is someone you share a grandparent with, but who isn't your sibling. Write the rule `cousin(X,Y)` and check that Gomez and Morticia are cousins.

Task 5. An aunt or uncle is your parent's sibling. Define the rule and query whose Uncle Fester is.

Task 6. Who are Pugsley's aunts and uncles? What about Debbie?

```
% Parent relationships
parent(debbie, sharron). parent(debbie, dave). parent(fester, eudora).
parent(fester, father). parent(gomez, eudora). parent(gomez, father).
...

% rules
grandparent(X,Z) :- parent(X,Y), parent(Y,Z).
sibling(X,Y) :- parent(X,Z), parent(Y,Z), X!=Y.
cousin(X,Y) :- grandparent(X,Z), grandparent(Y,Z), X!=Y, not sibling(X,Y).
aunt_or_uncle(X,Y) :- parent(X,Z), sibling(Y,Z).
```

Access control

Absolute basics

In your system you should have the following predicates:

- `file(X)` to state that `X` is a file.
- `principal(X)` to state that `X` is a principal (which is a fancy way of saying a *user*).
- `canRead(P, F)` to state that the principal `P` can read file `F`.

Discretionary access control

- All files have an `owner`. `owner(F,P)` states that principal `P` is the owner of file `F`.
- You can read a file if you are it's owner.
- You can read a file if `othersCanRead(F)` is true.

```
file(doc1). file(doc2). file(doc3).
principal(a). principal(b). principal(c).
owner(doc1, a). owner(doc2, b). owner(doc3, c).
%othersCanRead(doc1).

canRead(P, F) :- file(F), principal(P), owner(F, P).
canRead(P, F) :- file(F), principal(P), othersCanRead(F).
canRead(P, F) :- file(F), principal(P), owner(F, P0), saysCanRead(P0,P,F).
```

Delegated access control

- The owner of a file can state who else can read it. `saysCanRead(P1,P2,F)` states that `P1` says that `P2` can read file `F`. Remember to check that `P1` can only delegate access if they own `F`.

- A principal can delegate to other principals to decide who can read their files. `delegatesTo(P1,P2)` states that `P1` will allow `P2` to make access control decisions for them.
- Make sure that a principal whose been delegated to can also redelegate the decision to someone else!

Role-based access control

- Add roles to your access control system. For example, a principal may be happy to say that an auditor can read their files, but may not know who the auditor's currently are.
- `holds(P,R)` states that a principal `P` holds a role `R`. Update your `saysCanRead` rule to account for roles.

```
delegatesTo(b, c).
delegatesTo(c, e).
holds(a, user).
principal(d). principal(e).
holds(d, user). holds(e, common).

saysCanRead(P1, P2, F) :- saysCanRead(P1, P3, F), saysCanRead(P3, P2, F).
saysCanRead(P1, P2, F) :- owner(F, P1), holds(P1, R), holds(P2, R), P1!=P2, principal(P1), principal
(P2), file(F).
saysCanRead(P1, P2, F) :- owner(F, P1), delegatesTo(P1, P2), P1!=P2, principal(P1), principal(P2), fi
le(F).
saysCanRead(P1, P2, F) :- owner(F, P0), delegatesTo(P0, P1), delegatesTo(P1, P2), P0!=P1, P1!=P2, pri
ncipal(P0), principal(P1), principal(P2), file(F).
```

Mandatory access control

- Files have a security level.
  - `unclassified(F)` states the file is publicly available.
  - `secret(F)` states that the file is secret.
  - `topsecret(F)` states that the file is top secret.
- Security levels have an ordering
  - `unclassified` < `secret` < `topsecret`
- Principals have a clearance. `clearance(P,secret)` states that Principal `P` has a `secret` security clearance.

Implement the following security models:

Read down, write up *(Bell LaPadula)*

The *read down, write up* access control model is used to protect access to data.

- You can read a file if you have an appropriate or higher clearance than the file.
- You can write to a file if you have an appropriate or lesser clearance than the file (so that you can tell people with more clearance than you things without informing your peers).

```
unclassified(doc1).
secret(doc2).
topsecret(doc3).
file(doc4).
secret(doc4).
clearance(a,unclassified).
clearance(b,secret).
clearance(c,topsecret).
clearance(d,secret).
clearance(e,unclassified).

canRead(P,F) :- file(F), principal(P), unclassified(F).
canRead(P,F) :- file(F), principal(P), clearance(P, secret), secret(F).
canRead(P,F) :- file(F), principal(P), clearance(P, topsecret), secret(F).
canRead(P,F) :- file(F), principal(P), clearance(P, topsecret), topsecret(F).

canWrite(P,F) :- file(F), principal(P), topsecret(F).
canWrite(P,F) :- file(F), principal(P), clearance(P, secret), secret(F).
```

```
canWrite(P,F) :- file(F), principal(P), clearance(P, unclassified), secret(F).
canWrite(P,F) :- file(F), principal(P), clearance(P, unclassified), unclassified(F).
```