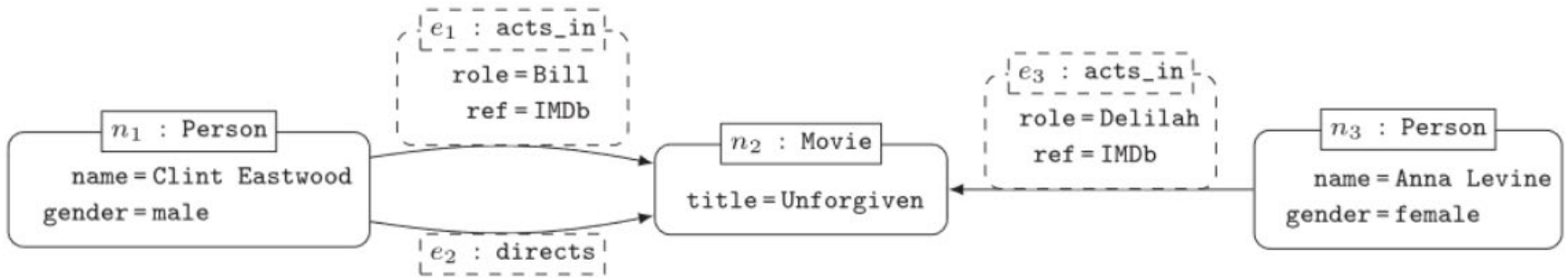
e_1 : acts_in
role = Bill
ref = IMDb
n_1 : Person
name = Clint Eastwood
gender = male
n_2 : Movie
title = Unforgiven
e_3 : acts_in
role = Delilah
ref = IMDb
n_3 : Person
name = Anna Levine
gender = female
e_2 : directs

Namespace (there is no need you define rdf, rdfs or owl prefixes, only those created by you)
mov =http://www.example.edu/movies/

SCHEMA

```
mov:directs  rdfs:domain  mov:director ;
             rdfs:range  mov:movie ;
             rdf:type  rdf:Property .

mov:name  rdfs:domain  mov:person ;
          rdfs:range  xsd:String ;
          rdf:type  rdfs:Property .

mov:gender  rdfs:domain  mov:person ;
            rdfs:range  xsd:String ;
            rdf:type  rdfs:Property .

mov:title  rdfs:domain  mov:movie ;
           rdfs:range  xsd:String ;
           rdf:type  rdfs:Property .

mov:role  rdfs:domain  mov:acts_in ;
          rdfs:range  mov:String ;
          rdf:type  rdfs:Property .

mov:ref  rdfs:domain  mov:acts_in ;
         rdfs:range  mov:String ;
         rdf:type  rdfs:Property .
```

SCHEMA (cont'd)

```
mov:acts_in_movie  rdfs:domain  mov:acts_in ;
                   rdfs:range  mov:movie ;
                   rdf:type  rdfs:Property .

mov:acts  rdfs:domain  mov:acts_in ;
          rdfs:range  mov:actor ;
          rdf:type  rdfs:Property .

mov:actor  rdfs:subClassOf  mov:person ;
           rdf:type  rdfs:Class .

mov:director  rdfs:subClassOf  mov:person ;
              rdf:type  rdfs:Class .

mov:person  rdf:type  rdfs:Class .
mov:acts_in  rdf:type  rdfs:Class .
mov:movie rdf:type rdfs:Class .
```

INSTANCES

```
mov:p1CE  mov:name "Clint Eastwood" ;
          mov:gender "male" ;
          rdf:type mov:person .

mov:p2AL  mov:name "Anna Levine" ;
          mov:gender "female" ;
          rdf:type mov:person .

mov:m1U  mov:title  "Unforgiven" ;
         rdf:type  mov:movie .

mov:acts_in01  mov:acts  mov:p1CE ;
               mov:acts_in_movie  mov:m1U ;
               mov:role  "Bill" ;
               mov:ref  "IMDb" ;
               rdf:type mov:acts_in .

mov:acts_in02  mov:acts  mov:p2AL ;
               mov:acts_in_movie  mov:m1U ;
               mov:role  "Delilah" ;
               mov:ref  "IMDb" ;
               rdf:type mov:acts_in .


mov:p1CE  mov:directs  mov:m1U .
```

Find all datatypes defined in the xsd vocabulary at: https://www.w3.org/2011/rdf-wg/wiki/XSD_Datatypes

Observations

- Realise that if we activate the RDFS regime entailment, most of the rdf:type explicitly stated in the previous RDFS graph **ARE NOT NEEDED** (they would be generated by inference instead)
  - mov:person, mov:director and mov:actor will be inferred as classes because they participate in triples with rdfs:subClassOf
  - All properties will be automatically asserted as properties when they participate in any triple defining its domain or range, or even when they participate in any instance triple
  - mov:p1CE, mov:p2AL, mov:m1U, mov:acts_in01 and mov:acts_in02 would be inferred as actors, director, movie or acts_in because they participate in triples whose properties have domain and range constraints

  In summary, all the red triples in the exercise can be omitted provided that we activate inference
- Realise mov:acts_in is an example of reification. Otherwise, we could not represent the attributes of the relationship (indeed, it is an n-ary relationship and would need a hyperedge)
- Realise that, for exemple, mov:p1CE will be inferred as actor and director, due to its participation in different triples whose properties constraint their respective domains. This is indeed correct, and a URI can be an instance of more than one class
- Nota that stating the domain of name / gender as follows:

```
mov:gender  rdfs:domain  mov:director , mov:actor .
```

Would imply that ANY triple whose predicate is mov:gender its subject will be automatically inferred to be of type mov:director **AND** mov:actor. You should not read this as an OR, because each triple generates its own inference. It is therefore preferable to assert the domain as type person (the superclass). This way, in the future, we could define other subsets of person (e.g., :artisticDirector rdfs:subClassOf :person) and still be fine, since the inference is generated for person. While in the other case, we would only generate the inference for :director and :actor. In this case, we would need to modify the above domain to include :artisticDirector explicitly. Bear this in mind, because this is the power of taxonomies and using superclasses instead of subclasses to type generic properties.