PSOA RuleML Integration of Relational and Object-Centered Geospatial Data

The 9th International Web Rule Symposium
RuleML 2015 Challenge
August 2-5, 2015

Gen Zou

Faculty of Computer Science,
University of New Brunswick, Fredericton, Canada

Outline

Outline

1. Background
2. Data Sets
3. Rules
4. Queries
5. Conclusion and Future Work

Backround

- Geospatial data sets have been increasingly available on the Web, e.g. Geonames and LinkedGeoData
- Many real-world applications are built on top of local data sets that contain geospatial information
- Integration of application data with external geospatial data can answer interesting geospatial queries

Backround

- Data can be modeled in different paradigms
  - Relational
    - Widely used for relational DBs and KBs, representing information in classical logic
  - Object-centered
    - Each object is represented by a unique Object IDentifier (OID) typed by a class and described by an unordered collection of slots, each being a pair of a name and a filler
  - Combined
- Integration needs cross-paradigm transformation, which can be expressed in the object-relational rule language PSOA RuleML

PSOA RuleML

- Integrates relational and object-centered modeling
- Generalizes F-logic, RIF-BLD, and POSL
- Uses **p**ositional-**s**lotted **o**bject-**a**pplicative (**psoa**) terms, permitting a relation application to have an OID – typed by the relation – and, orthogonally, its arguments to be positional or slotted

  General case (multi-tuple):

  $o\#f([t_{1,1} \ldots t_{1,n_1}] \ldots [t_{m,1} \ldots t_{m,n_m}]\ p_1\text{->}v_1 \ldots p_k\text{->}v_k)$

  Special cases (single-tuple brackets and zero-argument parentheses optional):

  Combined: $o\#f([t_1 \ldots t_n]\ p_1\text{->}v_1 \ldots p_k\text{->}v_k)$
  Positional: $o\#f([t_1 \ldots t_n])$
  Slotted: $o\#f(p_1\text{->}v_1 \ldots p_k\text{->}v_k)$
  Member-only: $o\#f()$

Outline

Data Sets

- Two relational data sets and one object-centered data set, expressed in PSOA RuleML presentation syntax
- Relational house rental data set

```
ex:HouseRentalInfo(1 "35 Routliffe Lane" "Toronto" "ON" "CA"
                     3 2500 "False"^^xs:boolean)
ex:HouseRentalInfo(2 "42 Frey Crescent" "Toronto" "ON" "CA"
                     2 900 "True"^^xs:boolean)
```

Arguments: ref number, street, city, province, country, number of bedrooms, price, furnished

Data Sets

- Relational data set containing addresses and their GPS coordinates (From online geocoding services)

```
gc:Geocode(43.778267 -79.426723
           "35 Routliffe Lane" "Toronto" "ON" "CA")
gc:Geocode(43.74242 -79.291529
           "42 Frey Crescent" "Toronto" "ON" "CA")
```

  Arguments: latitude, longitude, street, city, province, country

- Object-centered data set consisting of geospatial features (From Geonames)

```
<http://sws.geonames.org/9411373/>#gn:Feature(
    gn:name->"The Detour Store"
    gn:featureCode->gn:S.RET
    geo:lat->45.39748
    geo:long->-80.2468)
```

Outline

Hierarchy of Geospatial Entities

```
gr:SubwayStation##gr:GeoEntity
gr:Restaurant##gr:GeoEntity
gr:Store##gr:GeoEntity
gr:House##gr:GeoEntity
gr:HouseForRent##gr:House
```

- `gr:GeoEntity` class denotes all geospatial entities that can be located
- Every `gr:GeoEntity`-typed object has a slot `gr:coord` for the precise coordinates of its centroid

Integration Rules

- Map house rental data into objects of `gr:HouseForRent` subclass of `gr:GeoEntity` and extract address information

```
Forall ?Key ?Name ?Phone ?Street ?City ?Prov ?Country
       ?PostCode ?Addr
(
  Exists ?Addr
  (
    And(gr:HouseRentID(?RefNo)#gr:HouseForRent(
        ?Bedrooms ?Price ?Furnished gr:addr->?Addr)
        ?Addr#gr:Address(gr:street->?Street
                         gr:city->?City
                         gr:prov->?Prov
                         gr:country->?Country))
  )
  :- ex:HouseRentalInfo(?RefNo ?Street ?City ?Prov ?Country
                        ?Bedrooms ?Price ?Furnished)
)
```

Integration Rules

- Enrich each `GeoEntity` with a `gr:coord` slot, by retrieving the coordinates from `gc:Geocode` relation using its address

```
Forall ?O ?Ad ?Lat ?Long ?Street ?City ?Prov ?Country
(
  ?O#gr:GeoEntity(gr:coord->gr:Point(?Lat ?Long))
  :- And(?O#gr:GeoEntity(gr:addr->?Ad)
         ?Ad#gr:Address(gr:street->?Street
                        gr:city->?City
                        gr:prov->?Prov
                        gr:country->?Country)
         gc:Geocode(?Lat ?Long ?Street ?City ?Prov ?Country))
)
```

Integration Rules

Map objects from the object-centered data set into objects of `gr:GeoEntity`

```
Forall ?O ?Name ?Lat ?Long
(
  ?O#gr:GeoEntity(gr:name->?Name
                  gr:coord->gr:Point(?Lat ?Long))
    :- ?O#gn:Feature(gn:name->?Name
                     geo:lat->?Lat
                     geo:long->?Long)
)
```

Integration Rules

Map feature codes in the object-centered data set into corresponding `gr:GeoEntity` subclass

```
Forall ?O
(
   ?O#gr:SubwayStation
     :- ?O#gn:Feature(gn:featureCode->gn:S.MTRO)
)

Forall ?O
(
   ?O#gr:Restaurant
     :- ?O#gn:Feature(gn:featureCode->gn:S.REST)
)

Forall ?O
(
   ?O#gr:Store
     :- ?O#gn:Feature(gn:featureCode->gn:S.RET)
)
```

Geospatial Relationship Inference Rules

Derive a `GeoEntity ?O` is in an `?Area` by composing slot `gr:coord` and `gr:RCCProperPartOf` relation

```
Forall ?O ?Ad ?Pt ?Area
(
 ?O#gr:GeoEntity(gr:in->?Area)
   :- And(
        ?O#gr:GeoEntity(gr:coord->?Pt)
        gr:RCCProperPartOf(?Pt ?Area)
      )
)
```

Geospatial Relationship Inference Rules

Derive `gr:RCCProperPartOf` between a point and a box, defined by its minimum latitude, minimum longitude, maximum latitude, and maximum longitude, through arithmetic computation

```
Forall ?Lat ?Long ?LatMin ?LongMin ?LatMax ?LongMax (
  gr:RCCProperPartOf(gr:Point(?Lat ?Long)
                     gr:Box(?LatMin ?LongMin ?LatMax ?LongMax))
    :- And (
          External(pred:numeric-greater-than-or-equal(?Lat ?LatMin))
          External(pred:numeric-greater-than-or-equal(?Long ?LongMin))
          External(pred:numeric-less-than-or-equal(?Lat ?LatMax))
          External(pred:numeric-less-than-or-equal(?Long ?LongMax))
       )
)
```

Geospatial Relationship Inference Rules

Derive the distance (measured in km) of `?O1` and `?O2` to be less or equal than `?Distance`, using external function `gr:distanceLessEqual`

```
Forall ?Lat1 ?Long1 ?Lat2 ?Long2 ?Distance ?Name ?G ?F
(
  gr:inDistance(?O1 ?O2 ?Distance)
    :-
      And(
        ?O1#gr:GeoEntity(gr:coord->gr:Point(?Lat1 ?Long1))
        ?O2#gr:GeoEntity(gr:coord->gr:Point(?Lat2 ?Long2))
        External(
           gr:distanceLessEqual(?Lat1 ?Long1 ?Lat2 ?Long2 ?Distance)))
)
```

Outline

Queries

Look for certain type of geospatial entities in a region and their addresses

```
And(?H#gr:HouseForRent(gr:in->gr:Box(43 -80 44 -79) gr:addr->?Addr)
    ?Addr#gr:Address(gr:street->?Street))
```

Look for all geospatial entities near specific entities

- All stores within 5km of the house with reference number 2:

```
And(?S#gr:Store(gr:name->?Name)
    gr:inDistance(gr:HouseRentID(2) ?S 5))
```

- All houses within 2km of a subway station and the name of the station

```
And(?S#gr:SubwayStation(gr:name->?Name)
    ?H#gr:HouseForRent gr:inDistance(?H ?S 2))
```

Outline

1. Background
2. Data Sets
3. Rules
4. Queries
5. Conclusion and Future Work

Conclusion and Future Work

- Demonstrate the usefulness of PSOA rules for the integration of geospatial data modeled in different paradigms
- Similar approach can be applied to enrich other local data sets containing address information
- Future work
  - Expand KB with required ground facts imported from relational/graph databases
  - Evaluate reasoning performance on expanded KB using PSOATransRun engine