PSOA RuleML Integration of Relational and Object-Centered Geospatial Data

Gen Zou

Faculty of Computer Science,
University of New Brunswick, Fredericton, Canada
gen.zou@unb.ca

Abstract. In recent years, many geospatial data sets have become available on the Web. These data can be incorporated into real-world applications to answer advanced geospatial queries. In this paper, we present a use case to integrate a local data set with external geospatial data sets on the Web. The data sets are modeled in different paradigms – relational and object-centered. The integration uses Positional-Slotted Object-Applicative (PSOA) RuleML, which combines the relational and object-centered modeling paradigms for databases as well as knowledge bases (KBs).

1 Introduction

In recent years, many geospatial data sets, e.g. LinkedGeoData[1] [1] and GeoNames[2], become available on the Web. Since a large number of real-world applications are built on top of data sets that contain geospatial information, enriching these data with external geospatial knowledge on the Web become an interesting topic. The integration becomes more complex when the data sets are modeled in different paradigms, e.g. some use the relational paradigm built on top of predicate applications while others use the object-centered paradigm built on top of RDF triples. In this paper, we present a use case that integrates two relational data sets and one object-centered data set to answer interesting geospatial queries. The integration is done using the PSOA RuleML language, which integrates the relational and the object-centered paradigms for knowledge representation.

The paper is organized as follows: Section 2 introduces the basics of PSOA RuleML. Section 3 explains the data sets used for integration and give sample facts. Section 4 explains integration rules. Section 5 gives sample geospatial queries that can be answered after integration. Section 6 concludes the paper.

[1] http://linkedgeodata.org
[2] http://www.geonames.org

2 PSOA RuleML

PSOA RuleML [2] is an object-relational Web rule language that integrates relations and graphs via positional-slotted object-applicative (psoa)[3] terms, which have the general form

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

Here, an object is identified by an Object IDentifier (OID) `o` and described by (1) a class membership `o # f`, (2) a set of tupled arguments $\mathtt{[t_{i,1}\ \ldots\ t_{i,n_i}]}$, $i = 1, \ldots, m$, each being a sequence of terms, and (3) a set of slotted arguments $\mathtt{p_j\text{->}v_j}$, $j = 1, \ldots, k$ representing attribute-value pairs. Each slot can have one or more fillers (values). The OID as well as tuples and, orthogonally, slots in a psoa term are all optional. A psoa term can express untyped objects by treating them as being typed by the root class `f`=`Top`.

A more detailed introduction, with many examples leading to the semantics, can be found in [3].

3 Data Sets

The data sets employed in the use case consist of three parts, all of which are converted into PSOA RuleML presentation syntax.

1. A relational data set that contains house rental information, where the arguments denote the unique reference number of the house, the street name and number, city name, province/state name, country name, number of bedrooms, price per month, and whether it is furnished. Some facts are shown in the following.

```
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)
ex:HouseRentalInfo(3 "200 Hilda Avenue" "Toronto" "ON" "CA"
                     4 2000 "True"^^xs:boolean)
ex:HouseRentalInfo(4 "56 Wellington Street West" "Toronto" "ON" "CA"
                     4 1695 "False"^^xs:boolean)
ex:HouseRentalInfo(5 "1130 McAllister Avenue" "Ottawa" "ON" "CA"
                     2 1300 "False"^^xs:boolean)
```

2. A relational data set containing coordinates of addresses in WGS 84 geodetic longitude-latitude spatial reference system used by the Global Positioning System. Such data can be obtained using online geocoding services. The relational arguments represent the latitude, the longitude, the street address, the city name, the province name, and the country name. Some facts are shown in the following.

[3] We use the upper-cased “PSOA” as a qualifier for the language and the lower-cased “psoa” for its terms.

```
gc:Geocode(43.778267 -79.426723 "35 Routliffe Lane" "Toronto" "ON" "CA")
gc:Geocode(43.74242 -79.291529 "42 Frey Crescent" "Toronto" "ON" "CA")
gc:Geocode(43.7955 -79.429234 "200 Hilda Avenue" "Toronto" "ON" "CA")
gc:Geocode(43.645289 -79.389063 "56 Wellington Street West" "Toronto" "ON" "CA")
gc:Geocode(39.78373 -100.445882 "1130 McAllister Avenue" "Ottawa" "ON" "CA")
```

3. An object-centered data set, extracted from Geonames, that contains information about geospatial entities. Each object is of the class `gn:Feature` (geospatial feature) and described by slots `gn:name`, `geo:lat`, `geo:long`, and `gn:featureCode` for its name, WGS84 latitude, WGS84 longitude, and feature code. The feature code describes the type of geospatial feature, e.g. a store, a hotel, a city, etc. Following is a example fact.

```
<http://sws.geonames.org/123456/>#gn:Feature(gn:name->"Canadian Tire"
                                              gn:featureCode->gn:S.RET
                                              geo:lat->43.7
                                              geo:long->-79.1)
```

4 Integration Rules

In order to extract geospatial knowledge from multiple data sets and integrate them, we first create a vocabulary to describe the geospatial entities in our KB. The constant names are modeled using IRIs starting with the prefix `gr:`, which is a shortcut for `http://psoa.ruleml.org/GeospatialRules#`. The class `gr:GeoEntity` 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, where the slot value is expressed as a `gr:Point` application to the latitude and longitude floating-point values. It also has an `gr:addr` slot for its address of the class `gr:Address`. Subclasses of `gr:GeoEntity` in the KB include `gr:SubwayStation`, `gr:Store` and `gr:House`. `gr:HouseForRent` is a subclass of `gr:House`. The set of subclasses can be expanded based on application requirements.

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

The class `gr:Address`, which is used to type addresses objects, is described by slots `gr:street`, `gr:city`, `gr:prov`, and `gr:country`.

The following rule extracts address information from the house rental relation. The OID of the `gr:HouseForRent` object is an application of the function `gr:HouseRentID` to the reference number `?RefNo`, which uniquely determines the house. While the OID of the `gr:Address` object is an existentially quantified variable `?Addr` denoting a system-generated OID.

```
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)
)
```

The coordinate of a geospatial entity with an address can be retrieved from the `gc:Geocode` relation, using the following rule.

```
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))
)
```

The following rule derives that an object `?O` is `gr:in` an `?Area` if `?O` has an address `?Ad`, `?Ad` has coordinates `?Pt`, and `?Pt` is a proper part of `?Area`. The conclusion adds to `?O` a slot name `gr:in` whose filler, `?Area`, is the result of the condition query evaluation. The condition performs a composition of the slot named `gr:coord`, with filler `?Pt`, followed by the binary relation `gr:RCCProperPartOf`, leading to `?Area`.

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

The following rule derives a “proper part of” relation between a point and an rectangular area expressed as an `gr:Box` application to the minimum latitude, the minimum longitude, the maximum latitude, and the maximum longitude of the area, by comparing the coordinates of the point with the coordinates of the boundaries of the area.

```
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))
       )
)
```

Next we discuss the integration of the graph data set for other geospatial queries. The following rule maps the slot values of `gn:name`, `geo:lat`, and `geo:long` of `gn:Feature` into slot values of `gr:name` and `gr:coord` for `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)
)
```

The classes of these objects can be refined using their feature codes, as shown in the following rules.

```
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)
)
```

With both data sets expressed as `GeoEntity` objects, we can introduce the following rule, which derives the distance (measured in km) of `?O1` and `?O2` to be less or equal than `?Distance`, using the external function `gr:distanceLessEqual`, which computes whether the distance of two GPS points (`?Lat1, ?Long1`) and (`?Lat2, ?Long2`) is less or equal than `?Distance`.

```
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)))
)
```

5 Geospatial Queries

The first kind of query that can be asked is one type of geospatial entities in a region, such as all houses available for rent in a region:

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

The result binds `?H` to `gr:HouseRentID(1)` ... `gr:HouseRentID(4)`, representing houses 1 to 4 from the original KB.

Another kind of query is to look for all geospatial entities near a specific entity.

For example, we can ask queries such as 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))
```

We can also look for all houses for rent within a certain distance of a place, e.g. 2km within the subway station named `"Spadina"`.

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

6 Conclusion

In this paper, we show a PSOA RuleML-based integration of a local relational house rental data set, with external geocoding data set and Geonames. The integration enables the use of reasoning engines to answer advanced geospatial queries. The approach can be applied to similar local data sets that contain address information. In the use case, we demonstrate the usefulness of object-relational PSOA rules, e.g. for expressing transformation among the relational, the object-centered, or the combined modeling paradigms. Future work includes: (1) exploring the use of Region Connection Calculus rules in GeospatialRules KB [4] to answer more geospatial queries; (2) expand the KB with more facts to test the scalability of the PSOATransRun engine [5].

References

1. Stadler, C., Lehmann, J., Höffner, K., Auer, S.: LinkedGeoData: A core for a Web of spatial open data. Semantic Web **3**(4) (2012) 333–354
2. Boley, H.: A RIF-Style Semantics for RuleML-Integrated Positional-Slotted, Object-Applicative Rules. In: Proc. 5th International Symposium on Rules: Research Based and Industry Focused (RuleML-2011 Europe), Barcelona, Spain. Lecture Notes in Computer Science, Springer (July 2011) 194–211
3. Boley, H.: PSOA RuleML: Integrated object-relational data and rules. In: Reasoning Web. Springer (2015)
4. Zou, G.: GeospatialRules: A Datalog$^+$ RuleML Rulebase for Geospatial Reasoning. In Patkos, T., Wyner, A., Giurca, A., eds.: Challenge+DC@RuleML. Volume 1211 of CEUR Workshop Proceedings., CEUR-WS.org (2014)
5. Zou, G., Boley, H.: PSOA2Prolog: Object-Relational Rule Interoperation and Implementation by Translation from PSOA RuleML to ISO Prolog. In: Proc. 9th International Web Rule Symposium (RuleML 2015), Berlin, Germany. Lecture Notes in Computer Science, Springer (August 2015)