RDF (Resource Description Framework)

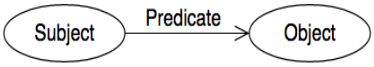


Fig. 1 An RDF graph with two nodes (Subject and Object) and a triple connecting them (Predicate)

Un triplet RDF est un statement RDF <*Sujet, Prédicat, Objet*>

- ***suject***, qui est une IRI ou une blank node
- ***prédicat***, qui est une IRI (Nature de la relation entre sujet et objet)
- ***objet***, qui est une IRI, un literal ou une blank node

Le *prédicat* dénote une propriété (relation binaire).

Un ensemble de triplets RDF représente un graphe RDF ou Suject & Objets sont des nodes du graphe RDF.
Il existe 3 types de nodes:

1. IRIs (Quelque chose dans le monde → **ressource (static referent)**)
2. Literals (Quelque chose dans le monde → **ressource (constant literal value**)
3. Blank nodes (Quelque chose existe dans la relation sans la nommer)

Vocabulaire RDF (RDFS language : Définition de vocabulaires pour définir les caractéristiques sémantiques des data RDF)

Class name	comment
rdfs:Resource	The class resource, everything.
rdfs:Literal	The class of literal values, e.g. textual strings and integers.
rdf:langString	**The class of language-tagged string literal values.**
rdf:HTML	**The class of HTML literal values.**
rdf:XMLLiteral	**The class of XML literal values.**
rdfs:Class	The class of classes.
rdf:Property	**The class of RDF properties.**
rdfs:Datatype	The class of RDF datatypes.
rdf:Statement	**The class of RDF statements.**
rdf:Bag	**The class of unordered containers.**
rdf:Seq	**The class of ordered containers.**
rdf:Alt	**The class of containers of alternatives.**
rdfs:Container	The class of RDF containers.
rdfs:ContainerMembershipProperty	The class of container membership properties, rdf:_1, rdf:_2, ..., all of which are sub-properties of 'member'.
rdf:List	**The class of RDF Lists.**

Property name	comment	domain	range
rdf:type	**The subject is an instance of a class.**	**rdfs:Resource**	**rdfs:Class**
rdfs:subClassOf	The subject is a subclass of a class.	rdfs:Class	rdfs:Class
rdfs:subPropertyOf	The subject is a subproperty of a property.	rdf:Property	rdf:Property
rdfs:domain	A domain of the subject property.	rdf:Property	rdfs:Class
rdfs:range	A range of the subject property.	rdf:Property	rdfs:Class
rdfs:label	A human-readable name for the subject.	rdfs:Resource	rdfs:Literal
rdfs:comment	A description of the subject resource.	rdfs:Resource	rdfs:Literal
rdfs:member	A member of the subject resource.	rdfs:Resource	rdfs:Resource
rdf:first	**The first item in the subject RDF list.**	**rdf:List**	**rdfs:Resource**
rdf:rest	**The rest of the subject RDF list after the first item.**	**rdf:List**	**rdf:List**
rdfs:seeAlso	Further information about the subject resource.	rdfs:Resource	rdfs:Resource
rdfs:isDefinedBy	The definition of the subject resource.	rdfs:Resource	rdfs:Resource
rdf:value	Idiomatic property used for structured values.	rdfs:Resource	rdfs:Resource

rdf:subject	The subject of the subject RDF statement.	rdf:Statement	rdfs:Resource
rdf:predicate	The predicate of the subject RDF statement.	rdf:Statement	rdfs:Resource
rdf:object	The object of the subject RDF statement.	rdf:Statement	rdfs:Resource

A partir de cette base, on peut définir un nouveau vocabulaire:

Construct	Syntactic form	Description
Class (a class)	**C** `rdf:type rdfs:Class`	**C** (a resource) is an RDF class
Property (a class)	**P** `rdf:type rdf:Property`	**P** (a resource) is an RDF property
type (a property)	**I** `rdf:type` **C**	**I** (a resource) is an instance of **C** (a class)
subClassOf (a property)	**C1** `rdfs:subClassOf` **C2**	**C1** (a class) is a subclass of **C2** (a class)
subPropertyOf (a property)	**P1** `rdfs:subPropertyOf` **P2**	**P1** (a property) is a sub-property of **P2** (a property)
domain (a property)	**P** `rdfs:domain` **C**	domain of **P** (a property) is **C** (a class)
range (a property)	**P** `rdfs:range` **C**	range of **P** (a property) is **C** (a class)

Exemple:

```
<Bob> <is a friend of> <Alice>.
```

Avec pour définition de "is a friend of" comme suit:

```
<Person> <type> <Class>
<is a friend of> <type> <Property>
<is a friend of> <domain> <Person>
<is a friend of> <range> <Person>
<is a good friend of> <subPropertyOf> <is a friend of>
```

Entailment regime / Extensions sémantiques

Donc de nouveaux vocabulaires sont possibles avec différents types de raisonnements associés (règles d'inférence à définir). Quand un type de raisonnement se trouve être utile pour beaucoup d'applications différentes ce type de raisonnement peut être documenté comme "entailment régime" [SPARQL11-ENTAILMENT] et [RDF11-MT]. Par exemple, RDFS est un entailment régime (ou une extension sémantique de RDF)

Graphes RDF multiples

RDF datasets (RDF 1.1) → Une collection de graphes RDF. Permet de travailler avec plusieurs graphes RDF tout en conservant leur contenu séparé.

- Un seul "*default graph*" (étant lui-même un graph RDF) **sans nom**
- Zéro ou plusieurs graphes nommés (qui sont des paires IRI/blank node ←→ graphe RDF). Les noms des graphes nommés sont uniques à l'intérieur d'un dataset.

Seul un de ces graphes a une IRI / blank node → **Graphes nommés**.
L'IRI ou blank not correspond au nom du graphe.
Note : SPARQL ne supporte pas des graphes RDF identifiés par des blank nodes.

Un triplet RDF encode un statement (expression logique) → un graphe RDF est une conjonction (ET logique) de ses triplets.

Relations entre graphes RDF:

1. **Implication** : A implique B, si A est vrai alors B est vrai aussi
2. **Equivalence** : A = B si A implique B et B implique A
3. **Inconsistance** : Le graphe contient une contradiction interne (Ne peut jamais être vérifié).

RDF literals datatypes

Core types	xsd:string	Character strings (but not all Unicode character strings)
	xsd:boolean	true, false
	xsd:decimal	Arbitrary-precision decimal numbers
	xsd:integer	Arbitrary-size integer numbers
IEEE floating-point numbers	xsd:double	64-bit floating point numbers incl. ±Inf, ±0, NaN
	xsd:float	32-bit floating point numbers incl. ±Inf, ±0, NaN
Time and date	xsd:date	Dates (yyyy-mm-dd) with or without timezone
	xsd:time	Times (hh:mm:ss.sss…) with or without timezone
	xsd:dateTime	Date and time with or without timezone
	xsd:dateTimeStamp	Date and time with required timezone (RDF 1.1)
Recurring and partial dates	xsd:gYear	Gregorian calendar year
	xsd:gMonth	Gregorian calendar month
	xsd:gDay	Gregorian calendar day of the month
	xsd:gYearMonth	Gregorian calendar year and month
	xsd:gMonthDay	Gregorian calendar month and day
	xsd:duration	Duration of time (RDF 1.1)
	xsd:yearMonthDuration	Duration of time (months and years only) (RDF 1.1)
	xsd:dayTimeDuration	Duration of time (days, hours, minutes, seconds only) (RDF 1.1)
Limited-range integer numbers	xsd:byte	-128…+127 (8 bit)
	xsd:short	-32768…+32767 (16 bit)
	xsd:int	-2147483648…+2147483647 (32 bit)
	xsd:long	-9223372036854775808…+9223372036854775807 (64 bit)
	xsd:unsignedByte	0…255 (8 bit)
	xsd:unsignedShort	0…65535 (16 bit)
	xsd:unsignedInt	0…4294967295 (32 bit)
	xsd:unsignedLong	0…18446744073709551615 (64 bit)
	xsd:positiveInteger	Integer numbers >0
	xsd:nonNegativeInteger	Integer numbers ≥0
	xsd:negativeInteger	Integer numbers <0
	xsd:nonPositiveInteger	Integer numbers ≤0
Encoded binary data	xsd:hexBinary	Hex-encoded binary data
	xsd:base64Binary	Base64-encoded binary data
Miscellaneous XSD types	xsd:anyURI	Absolute or relative URIs and IRIs
	xsd:language	Language tags per [BCP47]
	xsd:normalizedString	Whitespace-normalized strings
	xsd:token	Tokenized strings
	xsd:NMTOKEN	XML NMTOKENs
	xsd:Name	XML Names
	xsd:NCName	XML NCNames

OWL (Ontologie)

Utilisé pour processer des informations contenues dans des documents plutôt que juste les présenter aux utilisateurs.

RDF : Modele de donnée pour des ressources et les relations entre elles.
RDFS : Vocabulaire pour décrire les propriétés et les classes des ressources RDF
OWL : Ajoute plus de vocabulaire pour décrire les propriétés et les classes (relations entre classes, cardinalité, égalité,…). OWL-Full permet

3 sous-langages:

OWL-Lite : Hiérarchie de classification (taxonomie) et contraintes simples
OWL-DL : Expressivité max avec les caractéristiques de décidabilité (tous les calculs sont garantis d'être terminés en temps borné) et de completion (toutes les conclusions sont calculables). Utilisation des constructions du langage OWL avec certaines restrictions.
OWL-Full : Expressivité max + liberté syntaxique de RDF (Sans les caractéristiques de décidabilité et de completion). Permet d'augmenter la signification des vocabulaires prédéfinis (RDF/OWL). Il y a cependant peu de chance qu'un raisonneur puisse raisonner sur toutes les features de OWL-Full.

Eléments de langage OWL-Lite

RDF Schema Features:

- owl:Class (Thing, Nothing)
- rdfs:subClassOf
- rdf:Property
- rdfs:subPropertyOf
- rdfs:domain
- rdfs:range
- Individual

(In)Equality:

- owl:equivalentClass
- owl:equivalentProperty
- owl:sameAs
- owl:differentFrom
- owl:AllDifferent
- owl:distinctMembers

Property Characteristics:

- owl:ObjectProperty
- owl:DatatypeProperty
- owl:inverseOf
- owl:TransitiveProperty
- owl:SymmetricProperty
- owl:FunctionalProperty
- owl:InverseFunctionalProperty

Property Restrictions:

- owl:Restriction
- owl:onProperty
- owl:allValuesFrom
- owl:someValuesFrom

Restricted Cardinality:

- owl:minCardinality (only 0 or 1)
- owl:maxCardinality (only 0 or 1)
- owl:cardinality (only 0 or 1)

Header Information:

- owl:Ontology
- owl:imports

Class Intersection:

- owl:intersectionOf

Datatypes

- xsd datatypes

Versioning:

- owl:versionInfo
- owl:priorVersion
- owl:backwardCompatibleWith
- owl:incompatibleWith
- owl:DeprecatedClass
- owl:DeprecatedProperty

Annotation Properties:

- rdfs:label
- rdfs:comment
- rdfs:seeAlso
- rdfs:isDefinedBy
- owl:AnnotationProperty
- owl:OntologyProperty

Eléments de langage OWL-DL / OWL-Full

Class Axioms:

- *owl:oneOf owl:dataRange*
- *owl:disjointWith*
- *owl:equivalentClass* (applied to class expressions)
- *rdfs:subClassOf* (applied to class expressions)

Boolean Combinations of Class Expressions:

- *owl:unionOf*
- *owl:complementOf*
- *owl:intersectionOf*

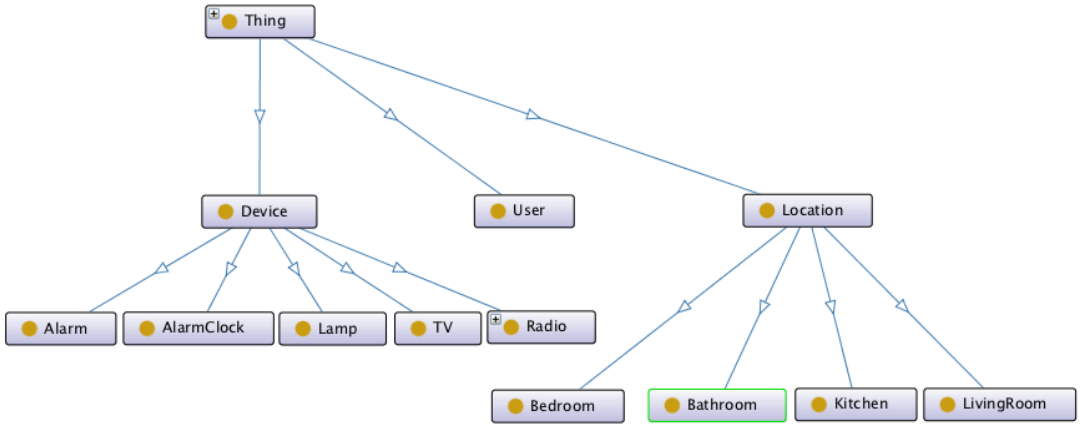
Arbitrary Cardinality:

- *owl:minCardinality*
- *owl:maxCardinality*
- *owl:cardinality*

Filler Information:

- *owl:hasValue*

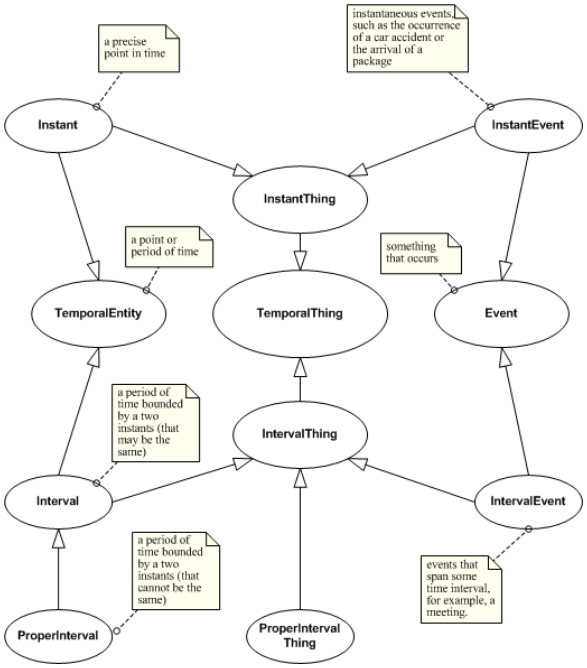
Exemple (Une ontologie simple)




```
<?xml version="1.0"?>
<!-- Namespaces -->
<rdf:RDF
    xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns="http://www.owl-ontologies.com/MyOntology1.owl#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xml:base="http://www.owl-ontologies.com/MyOntology1.owl">

<!-- Headers -->
  <owl:Ontology rdf:about=""/>

<!-- Elements -->
  <rdfs:Class rdf:ID="Location"/>
  <rdfs:Class rdf:ID="Device"/>
  <rdfs:Class rdf:ID="User"/>

<!-- Pre-defined devices -->

  <rdfs:Class rdf:ID="AlarmClock">
    <rdfs:subClassOf rdf:resource="#Device"/>
  </rdfs:Class>

  <rdfs:Class rdf:ID="Lamp">
    <rdfs:subClassOf rdf:resource="#Device"/>
  </rdfs:Class>

  <rdfs:Class rdf:ID="TV">
    <rdfs:subClassOf rdf:resource="#Device"/>
  </rdfs:Class>

  <rdfs:Class rdf:ID="Alarm">
    <rdfs:subClassOf rdf:resource="#Device"/>
  </rdfs:Class>

  <rdfs:Class rdf:ID="Radio">
    <rdfs:subClassOf rdf:resource="#Device"/>
  </rdfs:Class>

<!-- Pre-defined location -->
  <rdfs:Class rdf:ID="Kitchen">
    <rdfs:subClassOf rdf:resource="#Location"/>
  </rdfs:Class>

  <rdfs:Class rdf:ID="Bedroom">
    <rdfs:subClassOf rdf:resource="#Location"/>
  </rdfs:Class>

  <rdfs:Class rdf:ID="LivingRoom">
    <rdfs:subClassOf rdf:resource="#Location"/>
  </rdfs:Class>

  <rdfs:Class rdf:ID="Bathroom">
    <rdfs:subClassOf rdf:resource="#Location"/>
</rdfs:Class>

<!-- Added vocabulary -->

  <owl:ObjectProperty rdf:ID="IsIn">
    <rdfs:range rdf:resource="#Location"/>
    <rdfs:domain rdf:resource="#User"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:ID="hasLocation">
    <rdfs:range rdf:resource="#Location"/>
    <rdfs:domain rdf:resource="#Device"/>
  </owl:ObjectProperty>
</rdf:RDF>
```

Ajout de vocabulaire temporel (Ontologie OWL-Time) + 13 relations d'Allen (en bleu) mais sans règles d'inférence définies

Property Name	Domain	Range
before	TemporalEntity	TemporalEntity
after	TemporalEntity	TemporalEntity
hasBeginning	TemporalEntity	Instant
hasEnd	TemporalEntity	Instant
hasDurationDescription	TemporalEntity	DurationDescription
inside	Interval	Instant
intervalEquals	ProperInterval	ProperInterval
intervalBefore	ProperInterval	ProperInterval
intervalMeets	ProperInterval	ProperInterval
intervalOverlaps	ProperInterval	ProperInterval
intervalStarts	ProperInterval	ProperInterval
intervalDuring	ProperInterval	ProperInterval
intervalFinishes	ProperInterval	ProperInterval
intervalAfter	ProperInterval	ProperInterval
intervalMetBy	ProperInterval	ProperInterval
intervalOverlappedBy	ProperInterval	ProperInterval
intervalStartedBy	ProperInterval	ProperInterval
intervalContains	ProperInterval	ProperInterval
intervalFinishedBy	ProperInterval	ProperInterval
years	DurationDescription	xsd;decimal
months	DurationDescription	xsd;decimal
weeks	DurationDescription	xsd;decimal
days	DurationDescription	xsd;decimal
hours	DurationDescription	xsd;decimal
minutes	DurationDescription	xsd;decimal
seconds	DurationDescription	xsd;decimal
unitType	DateTimeDescription	TemporalUnit
year	DateTimeDescription	xsd;gYear
month	DateTimeDescription	xsd;gMonth
week	DateTimeDescription	xsd;nonNegativeInteger
day	DateTimeDescription	xsd;gDay
dayOfWeek	DateTimeDescription	DayOfWeek
dayOfYear	DateTimeDescription	xsd;nonNegativeInteger
hour	DateTimeDescription	xsd;nonNegativeInteger
minute	DateTimeDescription	xsd;nonNegativeInteger
second	DateTimeDescription	xsd;decimal
timeZone	DateTimeDescription	tzont;TimeZone
inDateTime	Instant	DateTimeDescription
inXSDDateTime	Instant	xsd;dateTime
hasDateTimeDescription	DateTimeInterval	DateTimeDescription
xsdDateTime	DateTimeInterval	xsd;dateTime



Représentation temporelle à partir de l'approche 4D-Fluents généralisée

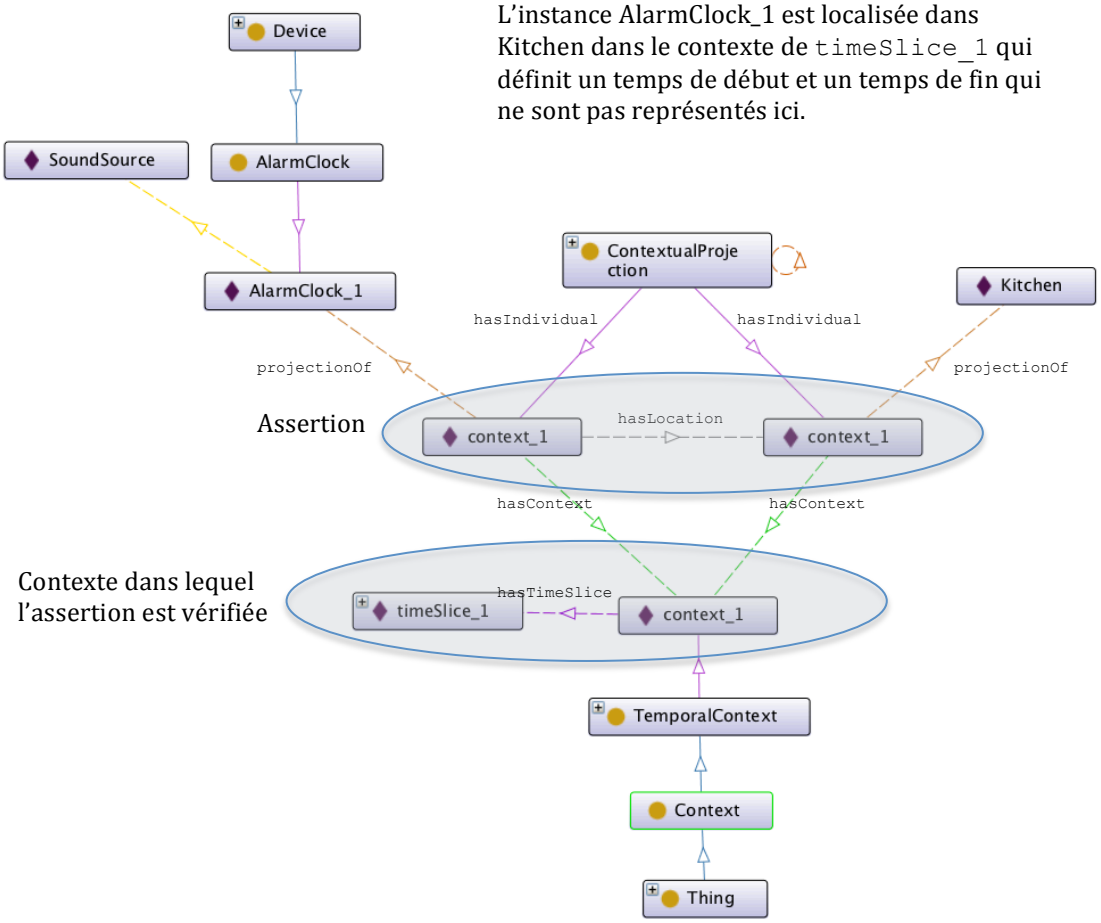
On suit l'approche présentée ici (Context slices):
http://ontologydesignpatterns.org/wiki/Submissions:Context_Slices

On définit une ontologie de gestion des contextes qui va être utilisée dans notre ontologie:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE rdf:RDF [
  <!ENTITY owl "http://www.w3.org/2002/07/owl#">
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
]>
<rdf:RDF xml:base=""
         xmlns:owl="&owl;"
         xmlns:rdf="&rdf;"
         xmlns:rdfs="&rdfs;">
  <owl:Ontology rdf:about="">
    <rdfs:label xml:lang="en">Context slices ontology logical  pattern</rdfs:label>
    <owl:versionInfo xml:lang="en">1.0</owl:versionInfo>
  </owl:Ontology>
  <owl:Class rdf:about="ContextualProjection">
    <rdfs:comment rdf:datatype="&xsd;string">A contextual projection or slice of an
entity or event.
The context defines the contextual extent of the slice.  If any relations hold in a
context, they
must be to other slices of the same context (there is no way to enforce this
constraint in OWL).
    </rdfs:comment>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        <owl:onProperty rdf:resource="#hasContext"/>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        <owl:onProperty rdf:resource="#projectionOf"/>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:allValuesFrom rdf:resource="Context"/>
        <owl:onProperty rdf:resource="#hasContext"/>
      </owl:Restriction>
    </rdfs:subClassOf>
    <owl:disjointWith rdf:resource="Context"/>
  </owl:Class>
  <owl:Class rdf:about="Context">
    <rdfs:comment rdf:datatype="&xsd;string">It is intended that this class should be
equated
with whatever class represents contexts, with possibly subclasses for different kinds
such as BeliefContext, TemporalInterval, etc.</rdfs:comment>
    <owl:disjointWith rdf:resource="#ContextualProjection"/>
  </owl:Class>
  <owl:ObjectProperty rdf:about="#contextualProperty">
    <rdfs:comment rdf:datatype="&xsd;string">A property that holds in a
context.</rdfs:comment>
    <rdfs:domain rdf:resource="#ContextualProjection"/>
    <rdfs:range rdf:resource="#ContextualProjection"/>
  </owl:ObjectProperty>
  <owl:FunctionalProperty rdf:about="#hasContext">
    <rdf:type rdf:resource="&owl;ObjectProperty"/>
    <rdfs:comment rdf:datatype="&xsd;string">The relation from a  ContextualProjection
to the context
in which some property holds</rdfs:comment>
    <rdfs:domain rdf:resource="#ContextualProjection"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:about="#projectionOf">
    <rdf:type rdf:resource="&owl;ObjectProperty"/>
    <rdfs:comment rdf:datatype="&xsd;string">The entity or event that a  context slice
is a slice of.</rdfs:comment>
    <rdfs:domain rdf:resource="#ContextualProjection"/>
  </owl:FunctionalProperty>
</rdf:RDF>
```

Scénario#1

On reprend l'ontologie précédente en stipulant que le radio réveil est localisé dans la cuisine entre 14h et 19h. **La propriété dynamique est ici la localisation des dispositifs**.

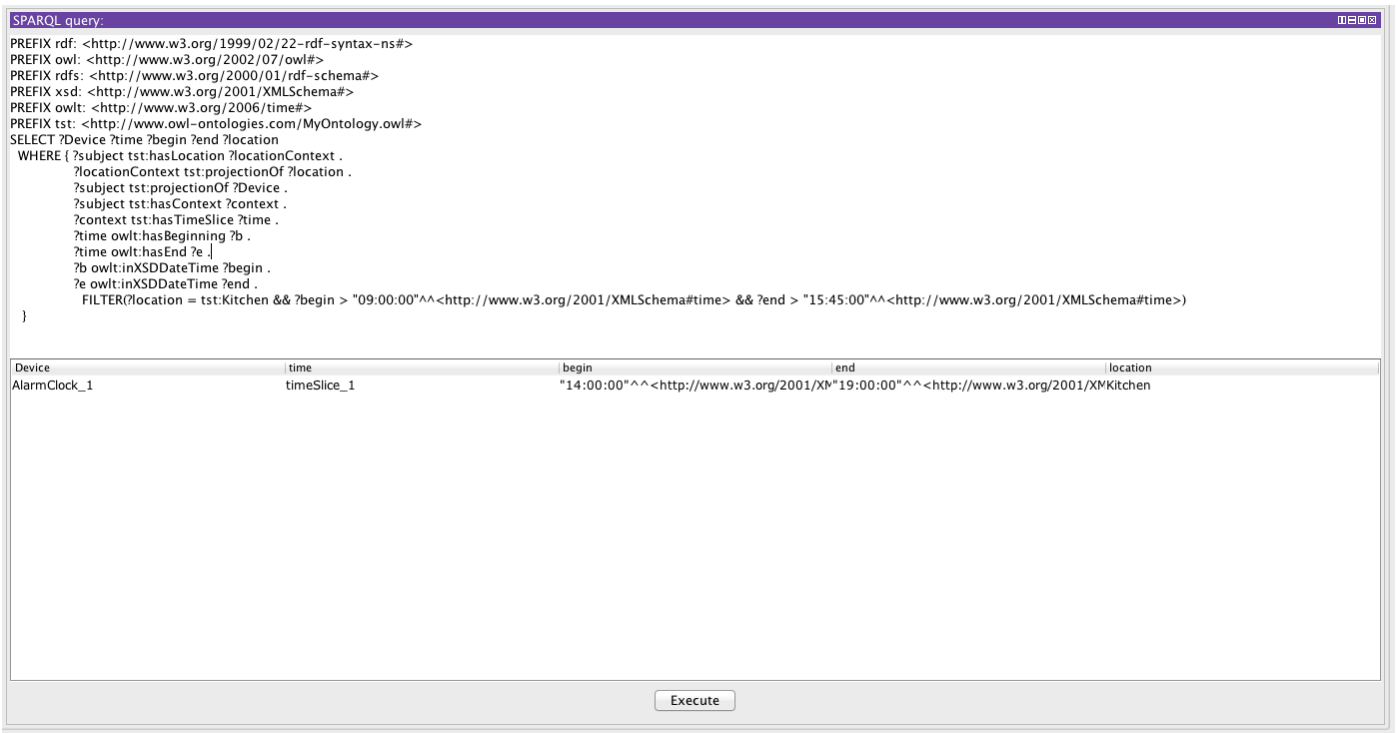


L'instance AlarmClock_1 est localisée dans Kitchen dans le contexte de `timeSlice_1` qui définit un temps de début et un temps de fin qui ne sont pas représentés ici.

```
…
<!-- Contextual property definition -->
<owl:ObjectProperty rdf:ID="hasLocation">
    <rdfs:subPropertyOf>
      <owl:ObjectProperty
rdf:ID="contextualProperty"/>
    </rdfs:subPropertyOf>
  </owl:ObjectProperty>
…
<!-- Contexts definition -->
  <AlarmClock rdf:ID="AlarmClock_1">
    <IsA rdf:resource="#SoundSource"/>
  </AlarmClock>
…
```

```
  …
  <owlt:Instant rdf:ID="i1">
    <owlt:inXSDDateTime
rdf:datatype="http://www.w3.org/2001/XMLSchema#time"
>14:00:00</owlt:inXSDDateTime>
  </owlt:Instant>
  <owlt:Instant rdf:ID="i2">
    <owlt:inXSDDateTime
rdf:datatype="http://www.w3.org/2001/XMLSchema#time"
>19:00:00</owlt:inXSDDateTime>
  </owlt:Instant>

  <owlt:ProperInterval rdf:ID="timeSlice_1">
    <owlt:hasBeginning rdf:resource="#i1"/>
    <owlt:hasEnd rdf:resource="#i2"/>
  </owlt:ProperInterval>

  <TemporalContext rdf:ID="context_1">
    <hasTimeSlice rdf:resource="#timeSlice_1"/>
  </TemporalContext>

  <ContextualProjection rdf:about="#Kitchen$context_1">
    <hasContext rdf:resource="#context_1"/>
    <projectionOf rdf:resource="#Kitchen"/>
  </ContextualProjection>
  <ContextualProjection rdf:about="#AlarmClock_1$context_1">
<hasContext rdf:resource="#context_1"/>
    <hasLocation rdf:resource="#Kitchen$context_1"/>
    <projectionOf rdf:resource="#AlarmClock_1"/>
  </ContextualProjection>
```

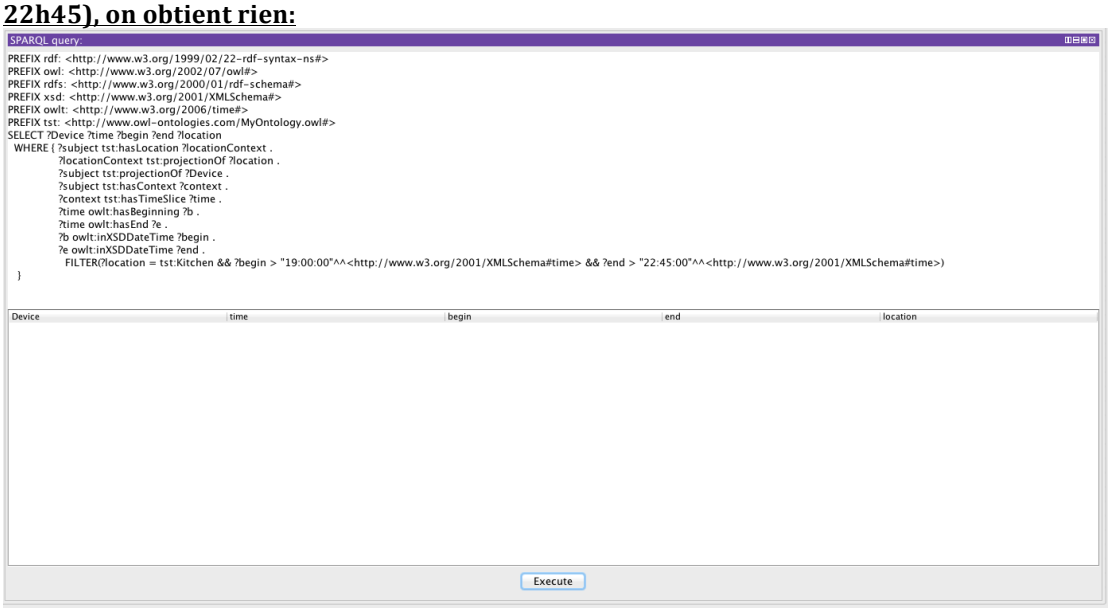
On peut alors faire une requête sparql pour extraire les dispositifs qui se trouvent à un endroit donné dans une intervalle de temps donnée (par exemple les dispositifs dans la cuisine):



On retrouve bien AlarmClock_1 qui est dans ce cas.

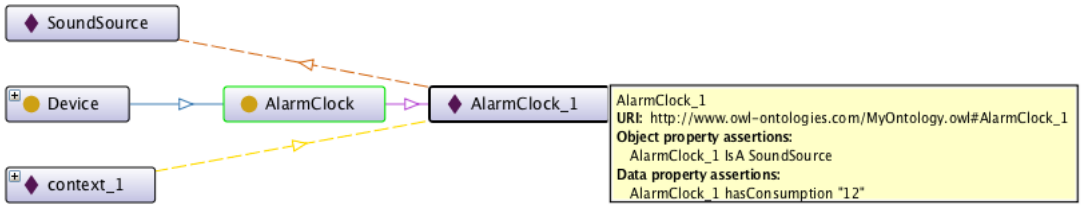
La requête est assez complexe mais il existe la fonction now() de SPARQL qui permet de retourner l'heure à laquelle est effectuée la requête. Il suffit de comparer les heures de start/end des timeslices avec l'heure retournée par now() pour avoir la liste des dispositifs disponibles au moment de la requête.
Cette fonction ne marche pas dans protégé ☹ je n'ai donc pas pu l'essayer.

Dans un autre contexte temporel ou aucun dispositif est valide (par ex. Entre 19h et 22h45), on obtient rien:

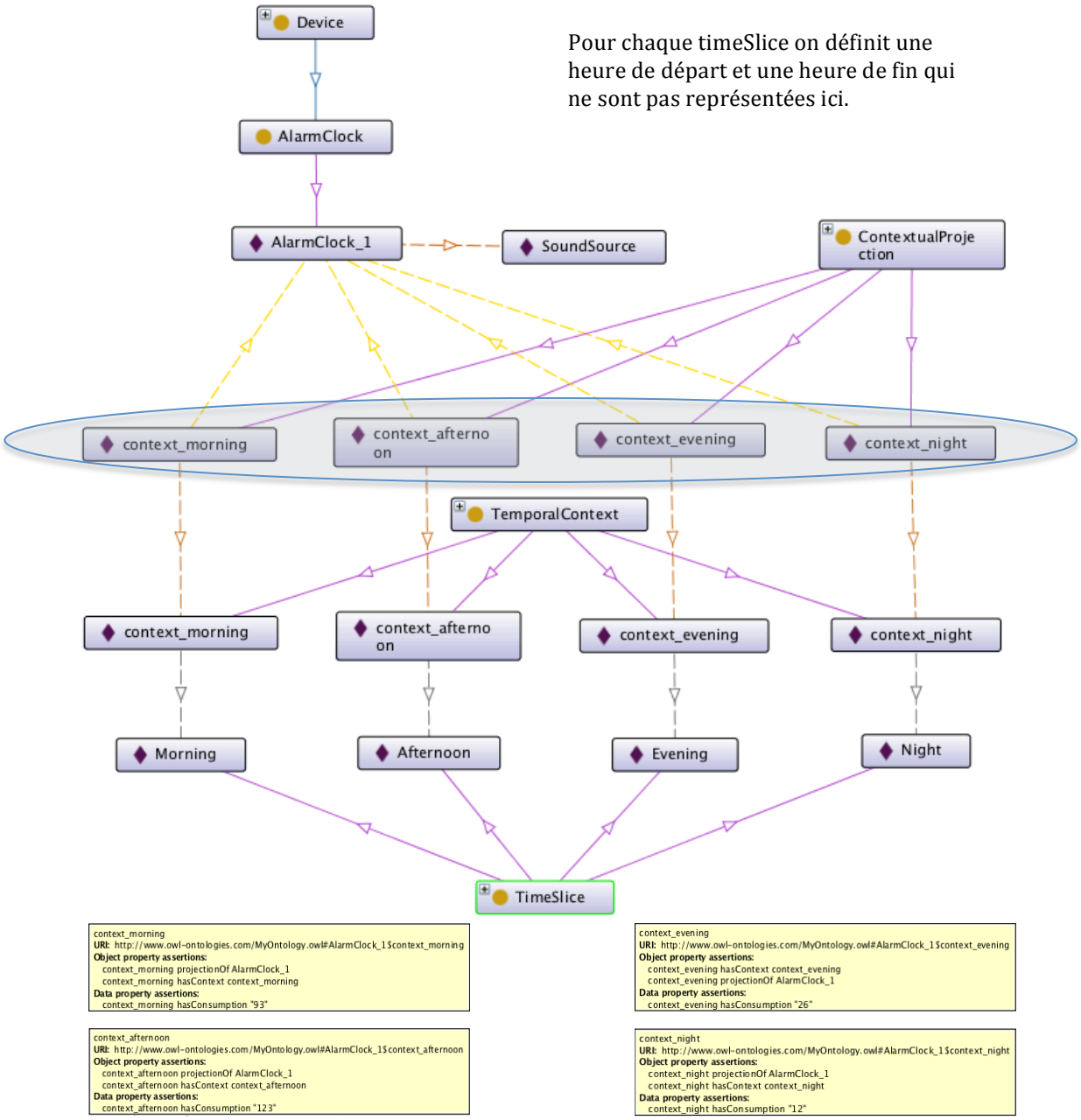


Scénario#2

Dans ce scénario, on considère que les dispositifs n'ont pas la même consommation électrique selon le moment dans la journée.
Par exemple, on peut considérer que le réveil (ok, c'est un peu idiot ☺) a une consommation électrique fixe (ici 12)



On peut très bien rendre la propriété 'hasConsommation' dynamique en fonction des moments dans la journée en la contextualisant:



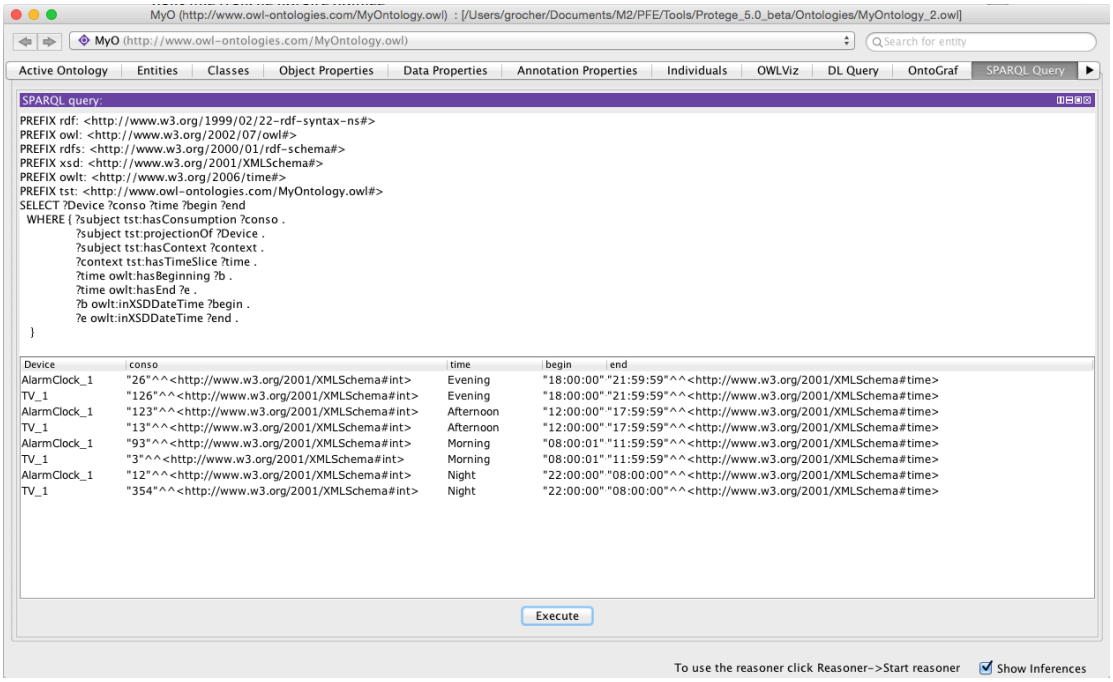
Pour chaque timeSlice on définit une heure de départ et une heure de fin qui ne sont pas représentées ici.

```
…
<!-- Contextual property definition -->
<owl:ObjectProperty rdf:ID="hasConsumption">
    <rdfs:range rdf:resource="&xsd;unsignedInt"/>
    <rdfs:subPropertyOf>
      <owl:ObjectProperty rdf:ID="contextualProperty"/>
    </rdfs:subPropertyOf>
</owl:ObjectProperty>
…
<!-- Pre-defined time slices -->

  <owlt:Instant rdf:ID="Night_start">
    <owlt:inXSDDateTime>22:00:00</owlt:inXSDDateTime>
  </owlt:Instant>
  <owlt:Instant rdf:ID="Night_end">
    <owlt:inXSDDateTime>08:00:00</owlt:inXSDDateTime>
  </owlt:Instant>

  <owlt:Instant rdf:ID="Morning_start">
    <owlt:inXSDDateTime>08:00:01</owlt:inXSDDateTime>
  </owlt:Instant>
  <owlt:Instant rdf:ID="Morning_end">
    <owlt:inXSDDateTime>11:59:59</owlt:inXSDDateTime>
  </owlt:Instant>

  <owlt:Instant rdf:ID="Afternoon_start">
    <owlt:inXSDDateTime>12:00:00</owlt:inXSDDateTime>
  </owlt:Instant>
  <owlt:Instant rdf:ID="Afternoon_end">
    <owlt:inXSDDateTime>17:59:59</owlt:inXSDDateTime>
  </owlt:Instant>

  <owlt:Instant rdf:ID="Evening_start">
    <owlt:inXSDDateTime>18:00:00</owlt:inXSDDateTime>
  </owlt:Instant>
  <owlt:Instant rdf:ID="Evening_end">
    <owlt:inXSDDateTime>21:59:59</owlt:inXSDDateTime>
  </owlt:Instant>

  <owlt:TimeSlice rdf:ID="Night">
    <owlt:hasBeginning rdf:resource="#Night_start"/>
    <owlt:hasEnd rdf:resource="#Night_end"/>
  </owlt:TimeSlice>

  <owlt:TimeSlice rdf:ID="Morning">
    <owlt:hasBeginning rdf:resource="#Morning_start"/>
    <owlt:hasEnd rdf:resource="#Morning_end"/>
  </owlt:TimeSlice>

  <owlt:TimeSlice rdf:ID="Afternoon">
    <owlt:hasBeginning rdf:resource="#Afternoon_start"/>
    <owlt:hasEnd rdf:resource="#Afternoon_end"/>
  </owlt:TimeSlice>

  <owlt:TimeSlice rdf:ID="Evening">
    <owlt:hasBeginning rdf:resource="#Evening_start"/>
    <owlt:hasEnd rdf:resource="#Evening_end"/>
  </owlt:TimeSlice>
…
```

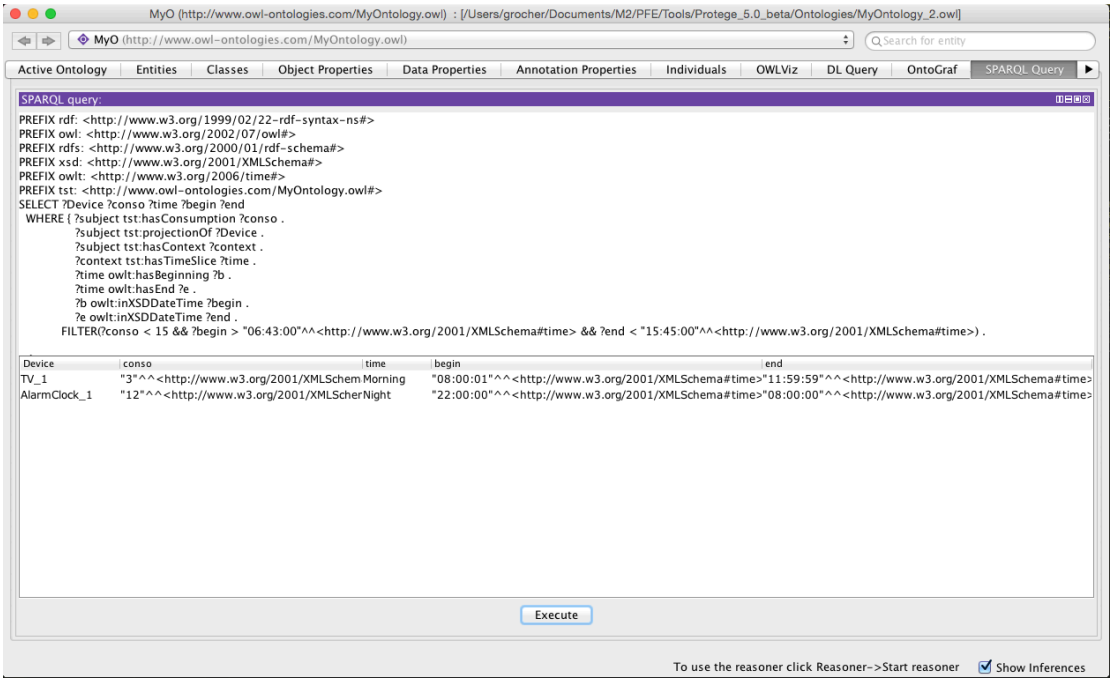
```
<!-- Contexts definition -->
<AlarmClock rdf:ID="AlarmClock_1">
    <IsA rdf:resource="#SoundSource"/>
  </AlarmClock>

  <TemporalContext rdf:ID="context_night">
    <hasTimeSlice rdf:resource="#Night"/>
  </TemporalContext>

  <TemporalContext rdf:ID="context_evening">
    <hasTimeSlice rdf:resource="#Evening"/>
  </TemporalContext>

  <TemporalContext rdf:ID="context_afternoon">
    <hasTimeSlice rdf:resource="#Afternoon"/>
  </TemporalContext>

  <TemporalContext rdf:ID="context_morning">
    <hasTimeSlice rdf:resource="#Morning"/>
  </TemporalContext>

  <ContextualProjection
rdf:about="#AlarmClock_1$context_night">
    <hasContext rdf:resource="#context_night"/>
    <hasConsumption>12</hasConsumption>
    <projectionOf rdf:resource="#AlarmClock_1"/>
  </ContextualProjection>

  <ContextualProjection
rdf:about="#AlarmClock_1$context_morning">
    <hasContext rdf:resource="#context_morning"/>
    <hasConsumption>93</hasConsumption>
    <projectionOf rdf:resource="#AlarmClock_1"/>
  </ContextualProjection>

  <ContextualProjection
rdf:about="#AlarmClock_1$context_afternoon">
    <hasContext rdf:resource="#context_afternoon"/>
    <hasConsumption>123</hasConsumption>
    <projectionOf rdf:resource="#AlarmClock_1"/>
  </ContextualProjection>

  <ContextualProjection
rdf:about="#AlarmClock_1$context_evening">
    <hasContext rdf:resource="#context_evening"/>
    <hasConsumption>26</hasConsumption>
    <projectionOf rdf:resource="#AlarmClock_1"/>
  </ContextualProjection>
```

On peut alors faire une requête SPARQL pour demander les dispositifs qui consomment le moins dans une tranche horaire donnée:

Par exemple, nous avons deux dispositifs qui consomment différemment en fonction de la tranche horaire (ontologie vue précédemment):

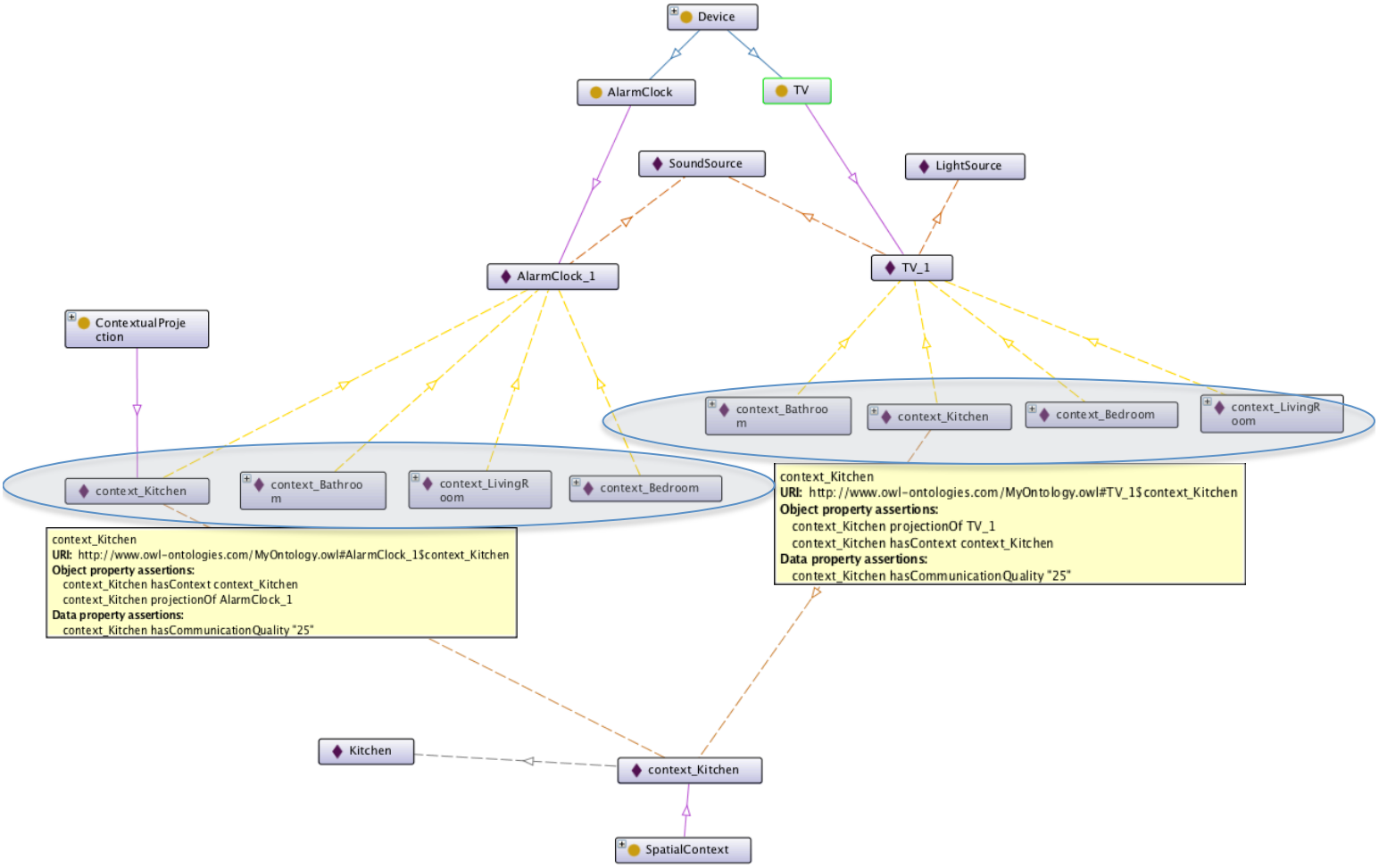


Maintenant on souhaite obtenir les dispositifs qui consomment moins de 15W entre 06h43 et 15h45:
Seulement 2 dispositifs correspondent :



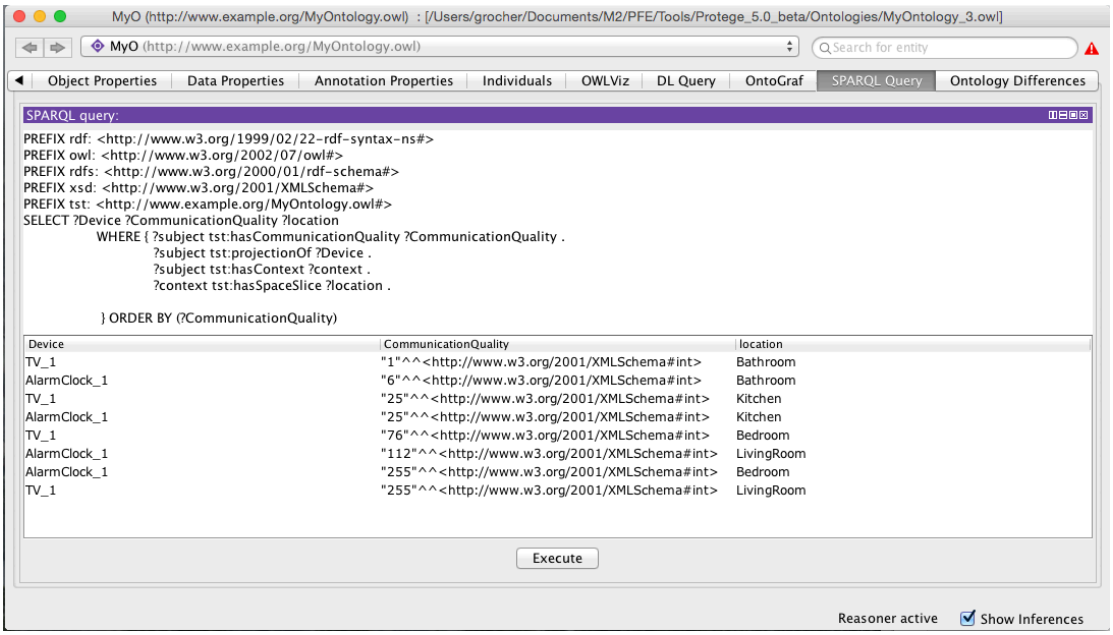
Scénario#3

Dans ce scénario on considère que le taux de qualité de communication avec les dispositifs est fonction de l'endroit ou ils se situent. La propriété dynamique va donc être la qualité de communication que l'on va contextualiser en fonction localisation des différents dispositifs: (Ca permet de faire des requetes du type : "*Je suis dans le salon, donne moi les services avec lesquels la qualité de communication est > 23*").



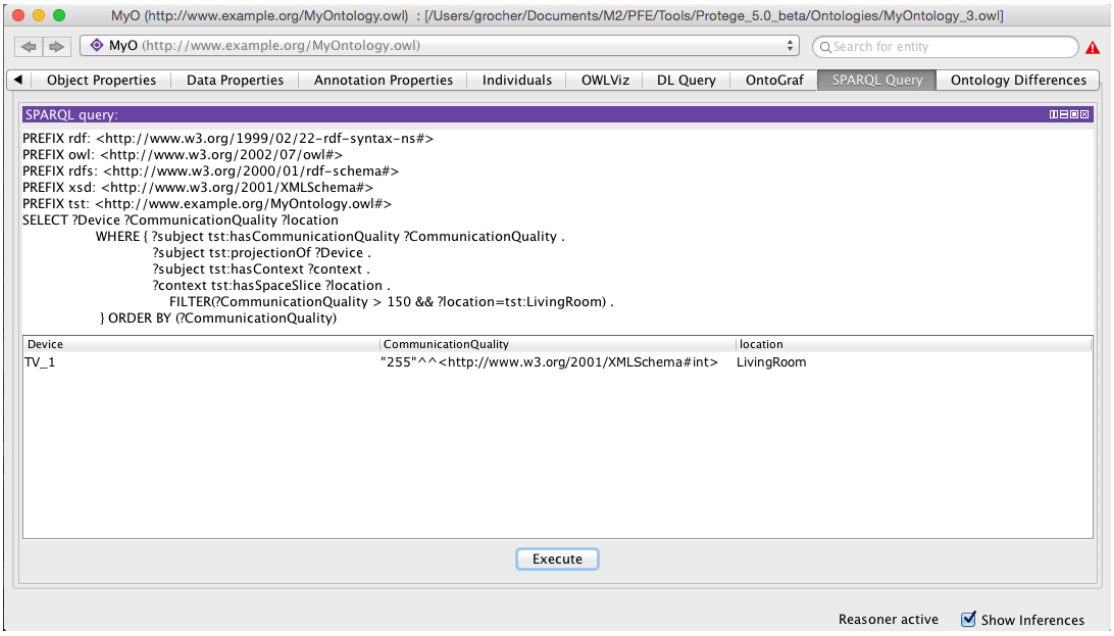
```
…
<!-- Contextual property definition -->
  <owl:ObjectProperty rdf:ID="hasCommunicationQuality">
    <rdfs:range rdf:resource="&xsd;unsignedByte"/>
    <rdfs:subPropertyOf>
      <owl:ObjectProperty rdf:ID="contextualProperty"/>
    </rdfs:subPropertyOf>
  </owl:ObjectProperty>…

  <owl:ObjectProperty rdf:ID="hasSpaceSlice">
    <rdfs:range rdf:resource="#SpaceSlice"/>
    <rdfs:domain rdf:resource="#SpatialContext"/>
  </owl:ObjectProperty>
…
<!-- Contexts definition -->
<AlarmClock rdf:ID="AlarmClock_1">
    <IsA rdf:resource="#SoundSource"/>
  </AlarmClock>

  <TV rdf:ID="TV_1">
    <IsA rdf:resource="#SoundSource"/>
    <IsA rdf:resource="#LightSource"/>
  </TV>

  <SpatialContext rdf:ID="context_Kitchen">
    <hasSpaceSlice rdf:resource="#Kitchen"/>
  </SpatialContext>

  <SpatialContext rdf:ID="context_LivingRoom">
    <hasSpaceSlice rdf:resource="#LivingRoom"/>
  </SpatialContext>

  <SpatialContext rdf:ID="context_Bedroom">
    <hasSpaceSlice rdf:resource="#Bedroom"/>
  </SpatialContext>

  <SpatialContext rdf:ID="context_Bathroom">
    <hasSpaceSlice rdf:resource="#Bathroom"/>
  </SpatialContext>

  <ContextualProjection rdf:about="#AlarmClock_1$context_Kitchen">
    <hasContext rdf:resource="#context_Kitchen"/>
    <hasCommunicationQuality>25</hasCommunicationQuality>
    <projectionOf rdf:resource="#AlarmClock_1"/>
  </ContextualProjection>

  <ContextualProjection rdf:about="#AlarmClock_1$context_LivingRoom">
    <hasContext rdf:resource="#context_LivingRoom"/>
    <hasCommunicationQuality>112</hasCommunicationQuality>
    <projectionOf rdf:resource="#AlarmClock_1"/>
  </ContextualProjection>

  <ContextualProjection rdf:about="#AlarmClock_1$context_Bedroom">
    <hasContext rdf:resource="#context_Bedroom"/>
    <hasCommunicationQuality>255</hasCommunicationQuality>
    <projectionOf rdf:resource="#AlarmClock_1"/>
  </ContextualProjection>

  <ContextualProjection rdf:about="#AlarmClock_1$context_Bathroom">
    <hasContext rdf:resource="#context_Bathroom"/>
    <hasCommunicationQuality>6</hasCommunicationQuality>
    <projectionOf rdf:resource="#AlarmClock_1"/>
  </ContextualProjection>

  <ContextualProjection rdf:about="#TV_1$context_Kitchen">
    <hasContext rdf:resource="#context_Kitchen"/>
    <hasCommunicationQuality>25</hasCommunicationQuality>
    <projectionOf rdf:resource="#TV_1"/>
  </ContextualProjection>

  <ContextualProjection rdf:about="#TV_1$context_LivingRoom">
    <hasContext rdf:resource="#context_LivingRoom"/>
    <hasCommunicationQuality>255</hasCommunicationQuality>
    <projectionOf rdf:resource="#TV_1"/>
  </ContextualProjection>

  <ContextualProjection rdf:about="#TV_1$context_Bedroom">
    <hasContext rdf:resource="#context_Bedroom"/>
    <hasCommunicationQuality>76</hasCommunicationQuality>
    <projectionOf rdf:resource="#TV_1"/>
  </ContextualProjection>

  <ContextualProjection rdf:about="#TV_1$context_Bathroom">
    <hasContext rdf:resource="#context_Bathroom"/>
    <hasCommunicationQuality>1</hasCommunicationQuality>
     <projectionOf rdf:resource="#TV_1"/>
  </ContextualProjection>
```

On peut alors faire des requêtes SPARQL pour retourner tous les dispositifs et leur qualité de communication dans chacune des locations:



Ok, tout y est!

Maintenant, on sélectionne les dispositifs qui ont une qualité de communication > 150 dans le salon:



Seule la TV_1 répond à cette contrainte.

Appendix : Règles d'inférence

Les règles utilisent la convention suivante :

aaa, bbb, etc... : URI référence (tous les prédicats possibles d'un triplet)
uuu, vvv, etc... : Identifiant de ref. d'URI ou de blank node, (sujets possibles d'un triplet)
xxx, yyy etc... : URI référence, blank node identifier ou litéral (sujet ou objet d'un triplet)
lll : Un litéral;
_:nnn, etc... : Blank node identifiers.

Par défaut il existe les règles d'inférence suivantes:

literal generalization rule.

Rule Name	If E contains	then add
lg	uuu aaa lll .	uuu aaa _:nnn . where _:nnn identifies a blank node allocated to the literal lll by this rule.

literal instantiation rule.

Rule Name	If E contains	then add
gl	uuu aaa _:nnn . where _:nnn identifies a blank node allocated to the literal lll by rule lg.	uuu aaa lll .

RDF entailment rules

Rule Name	if E contains	then add
rdf1	uuu aaa yyy .	aaa rdf:type rdf:Property .
rdf2	uuu aaa lll . where lll is a well-typed XML literal .	_:nnn rdf:type rdf:XMLLiteral . where _:nnn identifies a blank node allocated to lll by rule lg.

RDFS entailment rules.

Rule Name	If E contains:	then add:
rdfs1	uuu aaa lll. where lll is a plain literal (with or without a language tag).	_:nnn rdf:type rdfs:Literal . where _:nnn identifies a blank node allocated to lll by rule lg.
rdfs2	aaa rdfs:domain xxx . uuu aaa yyy .	uuu rdf:type xxx .
rdfs3	aaa rdfs:range xxx . uuu aaa vvv .	vvv rdf:type xxx .
rdfs4a	uuu aaa xxx .	uuu rdf:type rdfs:Resource .
rdfs4b	uuu aaa vvv.	vvv rdf:type rdfs:Resource .
rdfs5	uuu rdfs:subPropertyOf vvv . vvv rdfs:subPropertyOf xxx .	uuu rdfs:subPropertyOf xxx .
rdfs6	uuu rdf:type rdf:Property .	uuu rdfs:subPropertyOf uuu .
rdfs7	aaa rdfs:subPropertyOf bbb . uuu aaa yyy .	uuu bbb yyy .
rdfs8	uuu rdf:type rdfs:Class .	uuu rdfs:subClassOf rdfs:Resource .
rdfs9	uuu rdfs:subClassOf xxx . vvv rdf:type uuu .	vvv rdf:type xxx .
rdfs10	uuu rdf:type rdfs:Class .	uuu rdfs:subClassOf uuu .
rdfs11	uuu rdfs:subClassOf vvv . vvv rdfs:subClassOf xxx .	uuu rdfs:subClassOf xxx .
rdfs12	uuu rdf:type rdfs:ContainerMembershipProperty .	uuu rdfs:subPropertyOf rdfs:member .
rdfs13	uuu rdf:type rdfs:Datatype .	uuu rdfs:subClassOf rdfs:Literal .

Additional inferences which would be valid under the extensional versions of the RDFS semantic conditions.

ext1	uuu rdfs:domain vvv .vvv rdfs:subClassOf zzz .	uuu rdfs:domain zzz .
ext2	uuu rdfs:range vvv .vvv rdfs:subClassOf zzz .	uuu rdfs:range zzz .
ext3	uuu rdfs:domain vvv . www rdfs:subPropertyOf uuu .	www rdfs:domain vvv .
ext4	uuu rdfs:range vvv . www rdfs:subPropertyOf uuu .	www rdfs:range vvv .
ext5	rdf:type rdfs:subPropertyOf www . www rdfs:domain vvv .	rdfs:Resource rdfs:subClassOf vvv .
ext6	rdfs:subClassOf rdfs:subPropertyOf www . www rdfs:domain vvv .	rdfs:Class rdfs:subClassOf vvv .
ext7	rdfs:subPropertyOf rdfs:subPropertyOf www . www rdfs:domain vvv .	rdf:Property rdfs:subClassOf vvv .
ext8	rdfs:subClassOf rdfs:subPropertyOf www . www rdfs:range vvv .	rdfs:Class rdfs:subClassOf vvv .
ext9	rdfs:subPropertyOf rdfs:subPropertyOf www . www rdfs:range vvv .	rdf:Property rdfs:subClassOf vvv .

rdfD1	ddd `rdf:type rdfs:Datatype .` uuu aaa "sss"^^ddd .	_:nnn `rdf:type` ddd . where _:nnn identifies a blank node allocated to "sss"^^ddd by rule rule lg.

rdfD2	ddd `rdf:type rdfs:Datatype .` uuu aaa "sss"^^ddd .	uuu aaa "ttt"^^ddd .

rdfD3	ddd `rdf:type rdfs:Datatype .` eee `rdf:type rdfs:Datatype .` uuu aaa "sss"^^ddd .	uuu aaa "ttt"^^eee .

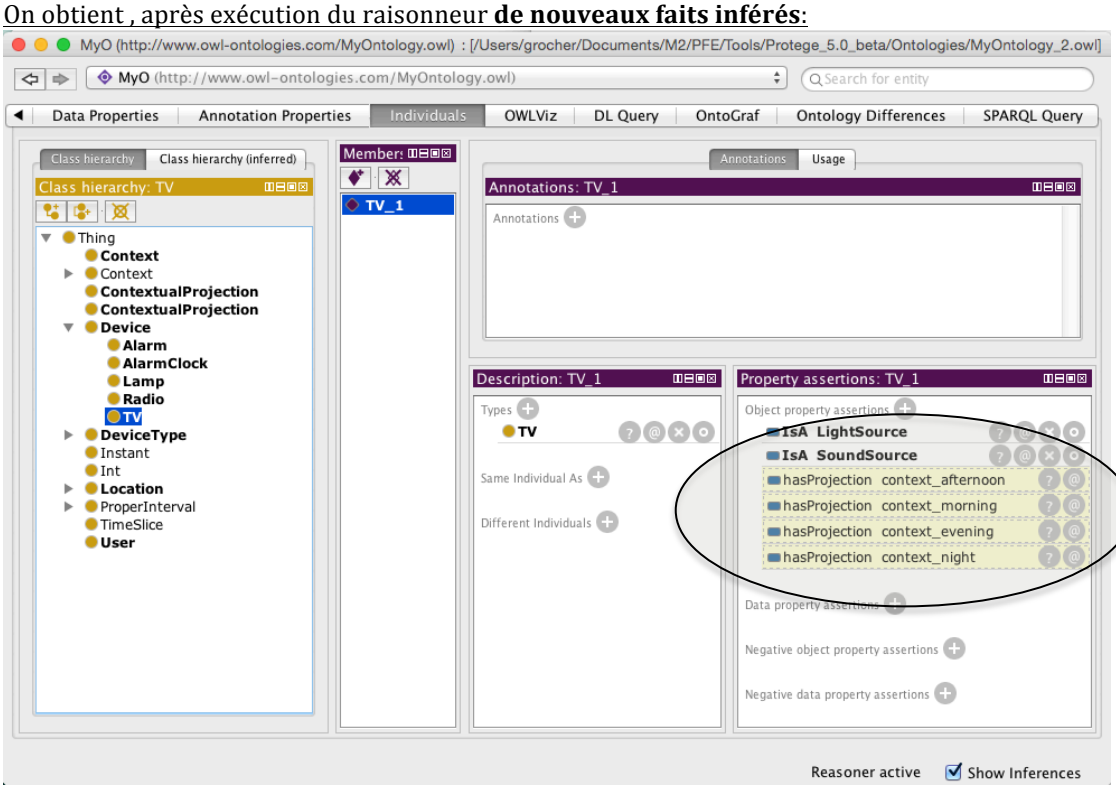
xsd 1a	uuu aaa "sss".	uuu aaa "sss"^^`xsd:string .`
xsd 1b	uuu aaa "sss"^^`xsd:string .`	uuu aaa "sss".

On peut ajouter des règles d'inférence 'custom' grâce à SWRL (Semantic Web Rule Langage). Par exemple, si on reprend le scenario 2 et qu'on souhaite inférer que si A *projectionOf* B alors B *hasProjection* A (exemple simple de symétrie mais qui permet de facilement retrouver tous les contextes d'un objet. A sa disparition, on efface les contextes (propriétés dynamiques), pas ses propriétés statiques). **Nécessite un raisonneur qui comprenne SWRL (Pellet par exemple)**

Modification de l'ontologie:

```
<swrl:Variable rdf:ID="x1"/>
<swrl:Variable rdf:ID="x2"/>
<swrl:Imp>
  <swrl:body rdf:parseType="Collection">
    <swrl:IndividualPropertyAtom>
      <swrl:propertyPredicate rdf:resource="#projectionOf"/>
      <swrl:argument1 rdf:resource="#x1"/>
      <swrl:argument2 rdf:resource="#x2"/>
    </swrl:IndividualPropertyAtom>
  </swrl:body>
  <swrl:head rdf:parseType="Collection">
    <swrl:IndividualPropertyAtom>
      <swrl:propertyPredicate rdf:resource="#hasProjection"/>
      <swrl:argument1 rdf:resource="#x2"/>
      <swrl:argument2 rdf:resource="#x1"/>
    </swrl:IndividualPropertyAtom>
  </swrl:head>
</swrl:Imp>
```

On obtient , après exécution du raisonneur **de nouveaux faits inférés**:



Etendre les fonctionnalités de SPARQL (Ajout d'opérateurs)

On se base sur Jena qui est utilisé dans Conquer.

https://www.google.fr/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&ved=0CCgQFjAB&url=https%3A%2F%2Fjena.apache.org%2Fdocumentation%2Fquery%2Fwriting_functions.html&ei=EDGEVKDrHsS2UY3SgRA&usg=AFQjCNHkm39QgmPdYsHNuwFXYZKerqVfVQ&sig2=EX9VG20T3OR8CtMwD1Gt6g&bvm=bv.80642063,d.d24