Ontology Patterns for Complex Activity Modelling

Georgios Meditskos, Stamatia Dasiopoulou, Vasiliki Efstathiou, Ioannis Kompatsiaris

Information Technologies Institute
Centre of Research & Technology – Hellas (CERTH)

Dementia Ambient Care: *Multi-Sensing Monitoring for Intelligent Remote Management and Decision Support*

www.demcare.eu

Outline

- **Ambient Assisted Living (AAL)**
- **Ontologies and Rules for Activity Recognition**
  - Obstacles to interoperability / reuse
- **Core Activity Pattern**
  - Composition and Specialisation
- **From Activity Patterns to SPARQL**
- **Conclusions / Future Directions**

Ambient Assisted Living: Key Challenge

- **Human Activity Recognition**
  - detect activities of daily living (ADL) for improving the healthcare support for elderly population
    - detect potentially dangerous behaviours
- **How**
  - multiple sensors are employed
    - e.g. contact sensors, cameras, microphones
  - collect and analyse data of different modalities
- **Why**
  - by combining different modalities we can infer more about the ***context***
    - any information that can be used to characterise the situation of an entity

Using Ontologies in AAL: Core Idea

- **Define formal models that are used to:**
  - Integrate low-level activities/events
    - Organise activities in hierarchies with properties, e.g. start/end times, agents/actors, temperature, light level, etc.
  - Model background knowledge specific to the domain
    - The structure and semantics of the complex activities that are built from atomic or other complex activities
- **High-level Interpretations**
  - Use of ontology reasoning (e.g. OWL DL reasoning)
- **Example (abstract syntax)**

MakeHotTea = Activity **and** (hasActor **only** (Person
and (uses **some** TeaBag) **and** (uses **some** Kettle)
and (inLocation **some** Kitchen)))

Limitations of Standard Ontology Semantics in AAL

- **A-temporal reasoning**
  - Complex activities are defined as the intersection of their constituent parts
  - Need for more flexible/expressive solutions
    - e.g. discrimination of sequential / interleaved activities
- **Reasoning about existing individuals**
  - Cannot assert new individuals for composite activities
    - Can only classify existing ones

Ontologies and Rules

- **Ontologies are combined with rules**
  - Handle the temporal extension
    - Custom functions, e.g. before, after, etc.
  - Express richer semantic relationships
    - Beyond tree-like relations
    - Generate new individuals
      - Caution: Need to handle termination problems

Example (unsafe rule!)

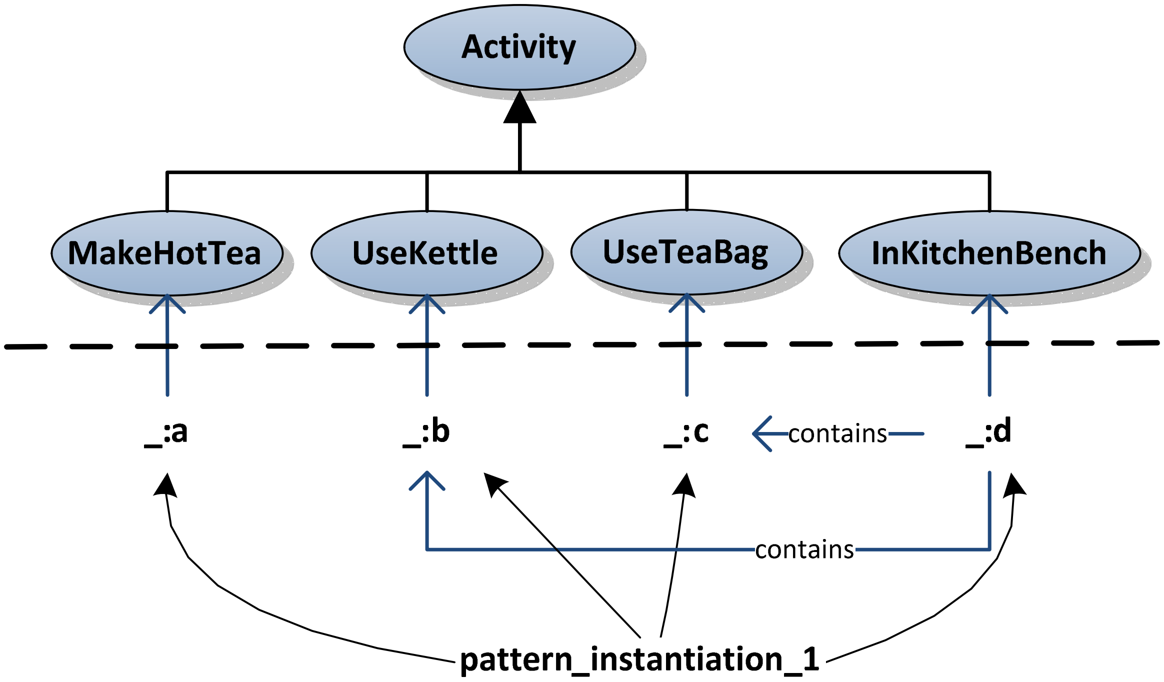
UseTeaBag(?u1), **UseKettle**(?u2), **NearKitchenBench**(?l),
contains(?l, ?u1), **contains**(?l, ?u2)
➔ **MakeHotTea**(?new), **start**(?new, ?l.start), **end**(?new, l.end)

Interoperability / Reuse Obstacles

- **The interpretation logic is defined outside the ontologies**
  - it is not part of the domain conceptual model
  - it is encapsulated in the implementation
    - e.g. in rules (or in any external module)
- **We cannot share knowledge relevant to activity recognition**
  - unless specific implementation details are made available
    - e.g. how new named individuals are asserted

Activity Patterns: Benefits

- **Formally capture the structure of complex activities**
  - Promote a well-defined description of patterns for detection
  - Achieve a high degree of interoperability
- **How**
  - Introduce a level of abstraction (vocabulary) for describing the context that defines complex activities
  - Need to define relations among classes (*meta-pattern*)



- **We propose**
  1. Core Activity Pattern (DnS/DUL)
  2. Two instantiations to handle different aspects
     - *Specialisation*
     - *Composition*
  3. Transformation procedure into SPARQL rules

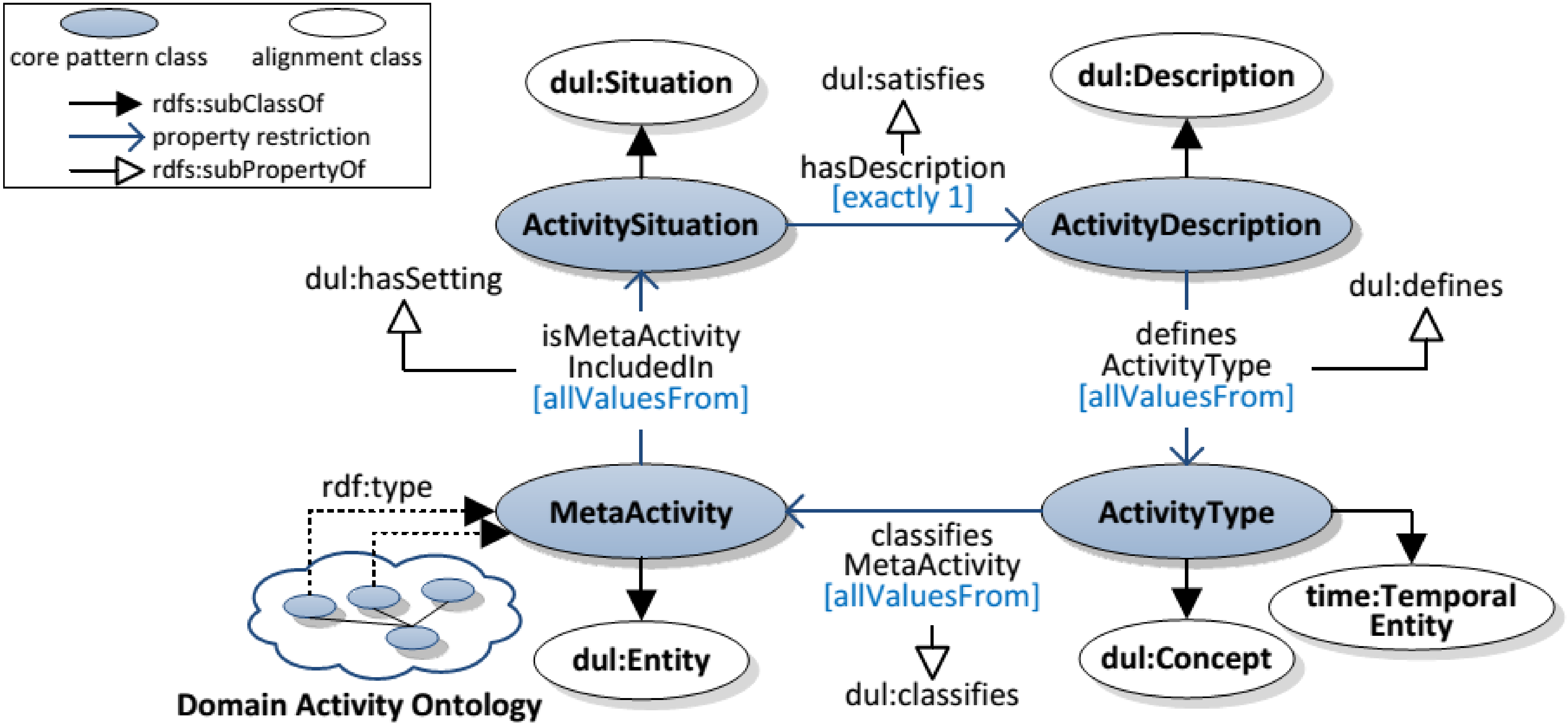
Core Activity Pattern

- **Vocabulary for defining contextualised views on complex activities in terms of**
  - the activity types that are involved
  - temporal relations among activity types
- **Treats domain activity classes as individuals**
  - Allows property assertions among activity types (*punning*)
- **Follows the conceptual model of DUL**
  - Specialisation of the Descriptions and Situations (DnS) pattern

Core Activity Pattern

- **Basic DnS building blocks**
  - **Situation**
    - A set of assertions
  - **Description**
    - Uses **DUL Concepts** to define interpretations (views) on Situations
- **Basic Core Activity Pattern building blocks**
  - **Activity Situation**: the set of domain activity classes that are involved in a instantiation
  - **Activity Description**: Creates a view on an Activity Situation by defining Activity Types and their relations (the context)
  - **Activity Types**: DUL Concepts that classify domain activity classes

Core Activity Pattern



Two Instantiations

- **Specialisation Pattern**
  - – Conceptual model for the specialisation of **existing** activity individuals in the activity domain hierarchy
    - define additional instance class membership relations
- **Composition Pattern**
  - – Conceptual model for the assertions of **new** activity instances

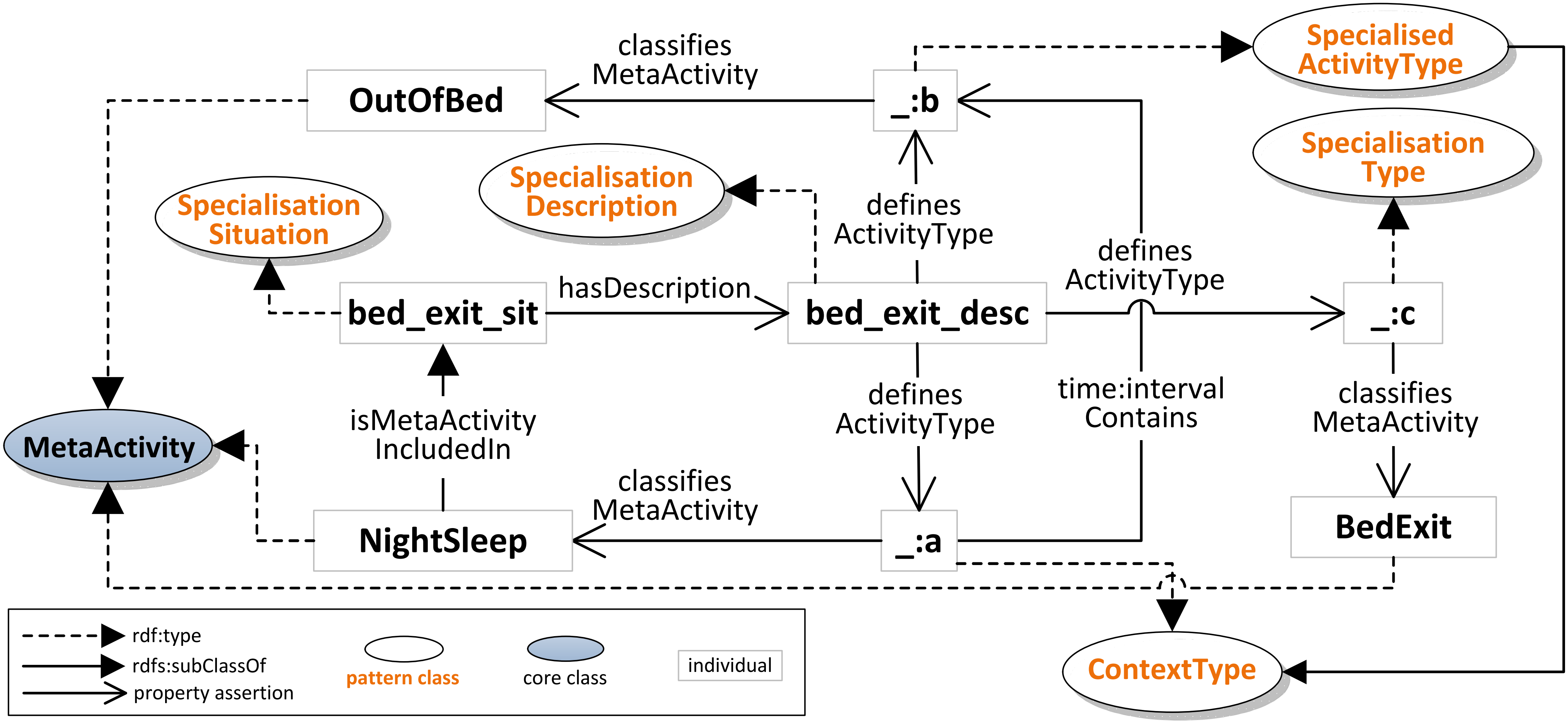
Specialisation Pattern

- **How an activity can be further specialised in the activity hierarchy**
  - Contextual dependencies
  - Temporal relations
- **Example**
  - "NightBedExit" activity
    - An "OutOfBed" activity that happens during a "NightSleep"
    - The "OutOfBed" is further specialised in the "NightBedExit" activity class

Specialisation Activity Types

- **ContextType**
  - Classifies the domain activity classes that comprise the activity context
  - e.g. OutOfBed, NightSleep
- **SpecialisedActivityType**
  - Classifies the domain activity class whose instance needs to be specialised
  - e.g. OutOfBed
- **SpecialisationType**
  - Classifies the domain activity class of the derived specialisation
  - e.g. BedExit

Bed Exit Example



- **ContextType:** OutOfBed, NightSleep
- **SpecialisedActivityType:** OutOfBed
- **SpecialisationType:** BedExit

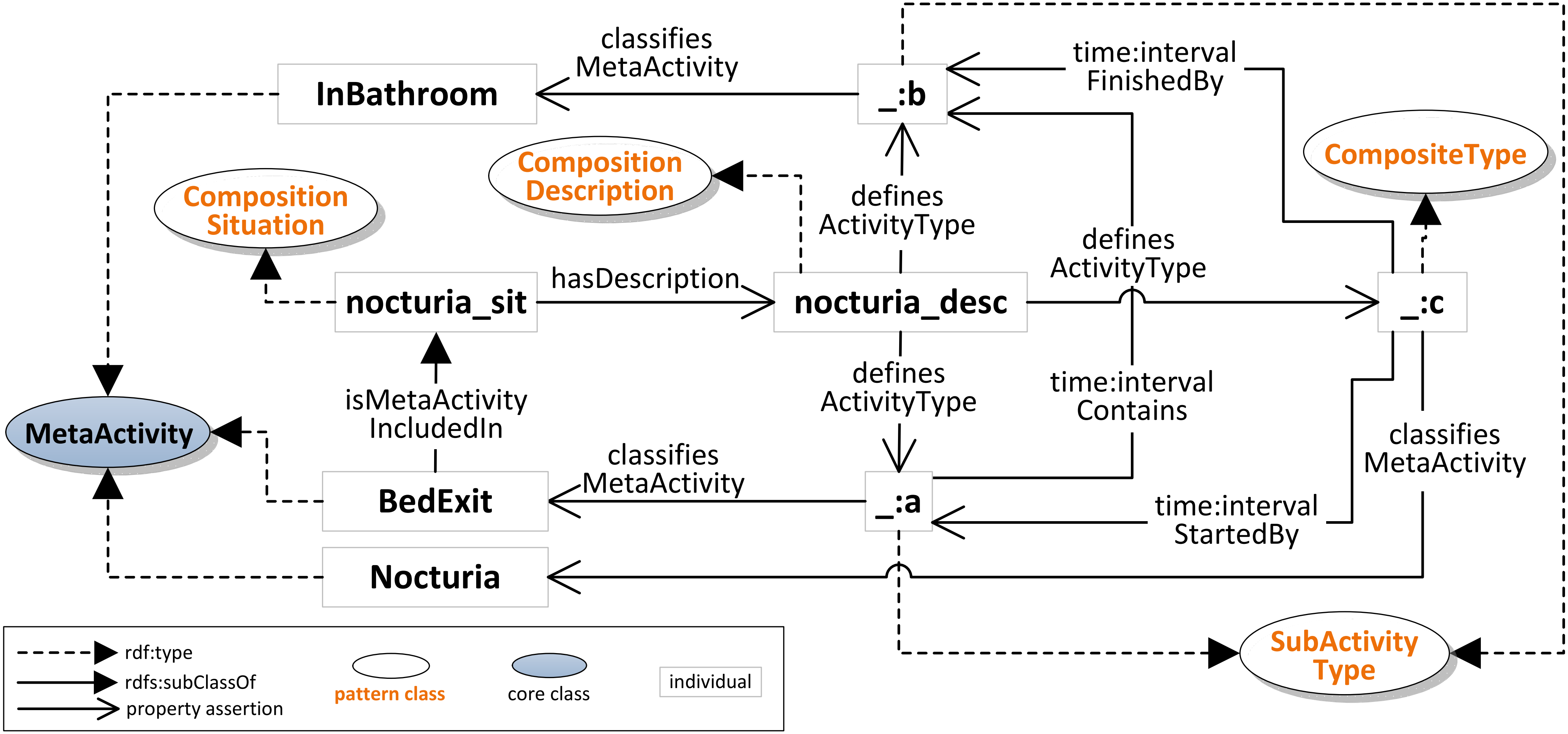
Composition Pattern

- **A new activity is derived based on**
  - The aggregation of other activities (sub-activities)
  - Temporal relations
- **Example**
  - "Nocturia" activity
    - When a "BedExit" activity contains an "InBathroom" activity
      - Neither the "BedExit" nor the "InBathroom" activity can be specialised as a "Nocturia" activity
    - Need to assert a new individual
      - start time: start time of BedExit
      - end time: end time of InBathroom

Composition Activity Types

- **CompositeType**
  - Classifies the complex activity to be inferred
  - e.g. Nocturia
- **SubActivityType**
  - Classifies the sub-activity classes
  - e.g. BedExit, InBathroom

Nocturia Example

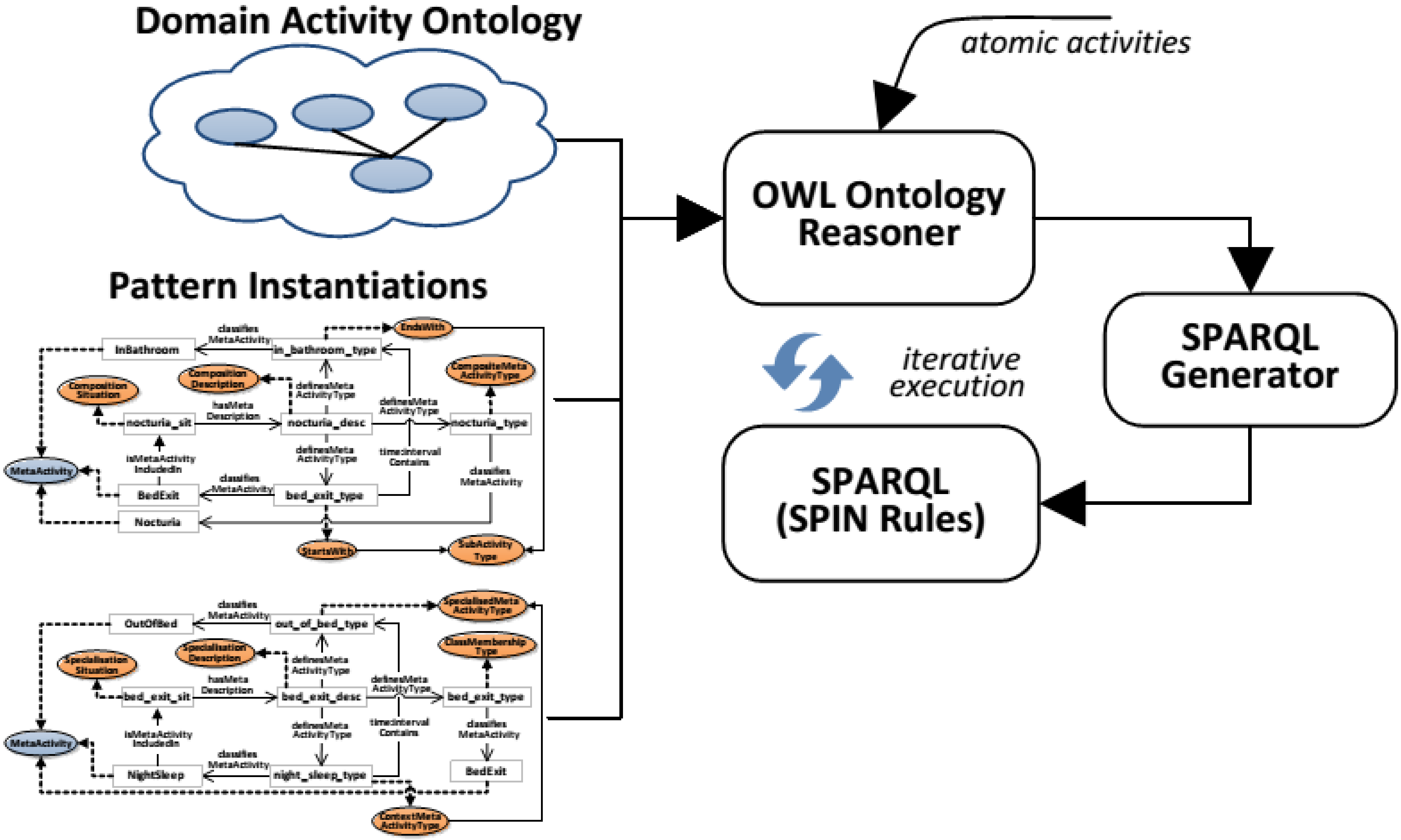


- **CompositeType:** Nocturia
- **SubActivityType:** BedExit, InBathroom

From Patterns to SPARQL

- **The patterns provide the structure and the semantics for activity detection**
  - The encapsulated semantics can be shared across applications with similar scopes
- **The way the semantics will be finally used depends on the implementation framework**
  - Rules (Jena, SPARQL, SWRL …)
  - Model behavioural profiles
    - compare behaviours
    - learn new behaviours
  - …
- **A proof-of-concept implementation**
  - A “compiler” for generating dynamic SPARQL rules (`CONSTRUCT` query graph patterns)

Architecture



Domain Activity Ontology

Pattern Instantiations

atomic activities

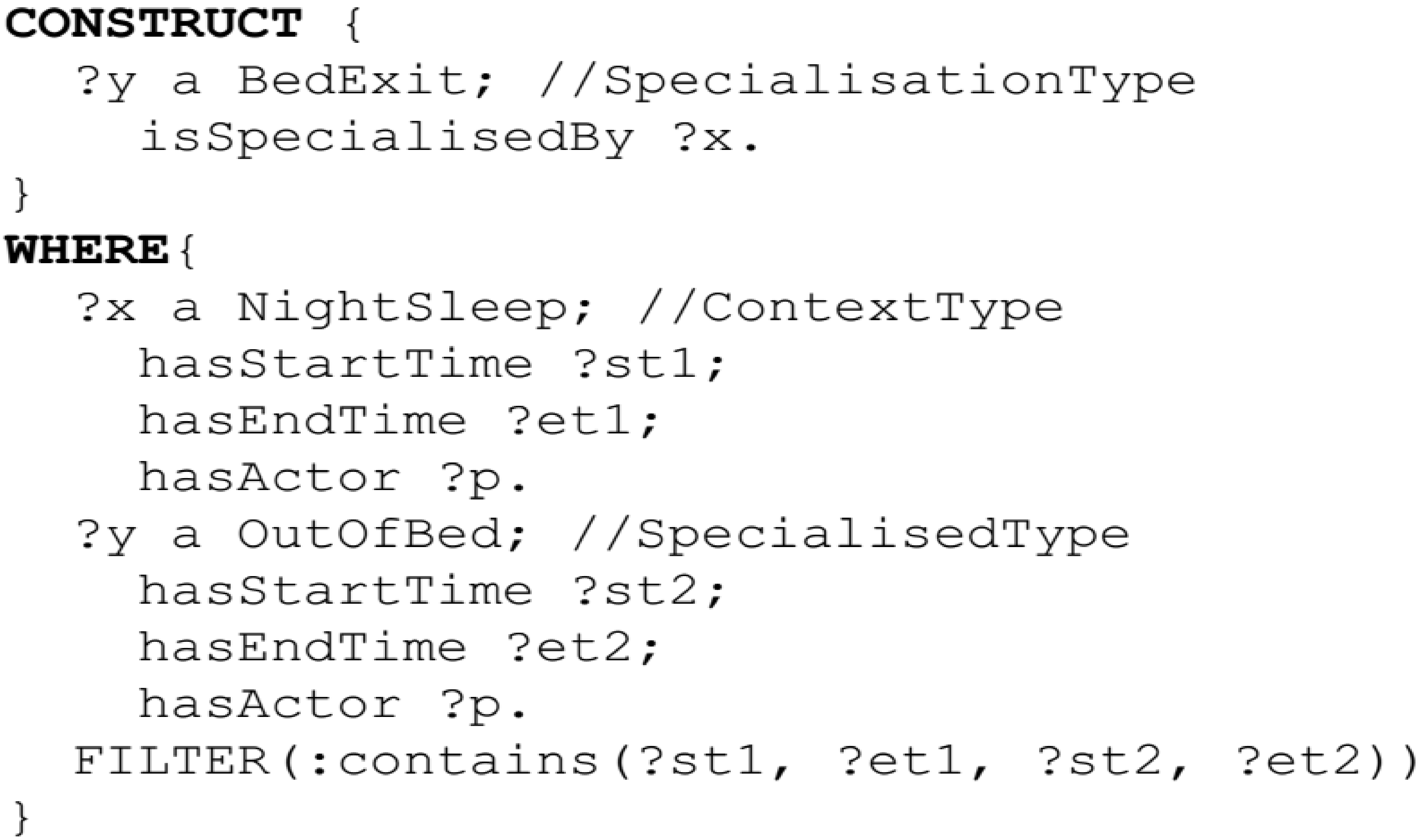
OWL Ontology Reasoner

SPARQL Generator

SPARQL (SPIN Rules)

iterative execution

BedExit SPARQL rule



```
CONSTRUCT {
  ?y a BedExit; //SpecialisationType
    isSpecialisedBy ?x.
}
WHERE{
  ?x a NightSleep; //ContextType
    hasStartTime ?st1;
    hasEndTime ?et1;
    hasActor ?p.
  ?y a OutOfBed; //SpecialisedType
    hasStartTime ?st2;
    hasEndTime ?et2;
    hasActor ?p.
  FILTER(:contains(?st1, ?et1, ?st2, ?et2))
}
```

Nocturia Composition Rule

```
CONSTRUCT {
  ?new a Nocturia; //CompositeType
    hasStartTime ?st1;
    hasEndTime ?et2;
    hasActor ?p;
    hasSubActivity ?x;
    hasSubActivity ?y.
}
WHERE{
  ?x a BedExit; //SubActivityType
    hasStartTime ?st1;
    hasEndTime ?et1;
    hasActor ?p.
  ?y a InBathroom; //SubActivityType
    hasStartTime ?st2;
    hasEndTime ?et2;
    hasActor ?p.
  FILTER(:contains(?st1, ?et1, ?st2, ?et2))
  BIND(:newURI(?x, ?y) as ?new)
}
```

Always returns the same URI for the same pair of ?x and ?y

Conclusions

- **Allow the formal representation of activity interpretation models**
  - Contextual dependencies
  - Temporal relations
- **Core Activity Pattern**
  - Extension of the DnS implementation in DUL
  - Two instantiations
    - Specialisation
    - Composition
- **Implementation using dynamically generated SPARQL rules**

Future Directions

- **Enhance the semantics of the activity patterns to conceptually represent**
  - Cardinality conditions
    - e.g. more than 2
  - Negation (NAF)
  - The presence of an activity type
    - e.g. EXISTS
  - Spatial relations
- **Provide an API for pattern instantiations**
  - Sesame, Jena
  - Pattern transformations