Using Domain Specific Hierachical Good Practice for Ranking Service Composition

Anca Marginean, Ioan Alfred Letia, Sergiu Zaporojan

September 22, 2014

Outline

Notation N3 and N3Logic

symbol	ex:book
variable	?a
universal variable	@forall :x, :y
existential variable	@forsome :x, :y
statement	ex:product ex:price 100
blank node	[ex:name "George"]
quoted formula	{ex:product ex:hasPrice 100}
statement with quoted formula	ex:client :says { :product :hasPrice :high}

Table: N3 elements

N3 descriptions

Example of offer in N3 notation

```
@prefix gr: <http://purl.org/goodrelations/v1#> .
ex:offer
    a gr:Offering ;
    gr:hasBusinessFunction gr:LeaseOut;
    gr:hasPriceSpecification ex:rental_price1,
                             ex:rental_price2.
```

Price description

```
ex:rental_price1
    a  gr:UnitPriceSpecification ;
    gr:hasCurrency "USD"^^xsd:string ;
    gr:hasCurrencyValue "2"^^xsd:float ;
    gr:hasUnitOfMeasurement "DAY"^^xsd:string ;
    gr:valueAddedTaxIncluded "true"^^xsd:boolean .
```

N3 rules

- Example of rules

```
{  ?p a ex:ExpertReader .
   ?p ex:says  {?book ex:quality ex:high} }
     => { ?book ex:quality ex:high }    (1)

{  ?book ex:quality ex:high.
   ?book ex:quality ex:low}
     =>  false.     (2)

 ex:john ex:hasFriend _:f.   (3)
```

- Reasoners: CWM, Euler (EYE)

RESTdesc - Formal Representation of Services

```
@prefix exF: <http://myflickerApi.org/photo#>.
@prefix foaf: <http://xmlns.com/foaf/0.1/>
@prefix tmpl: <http://purl.org/restdesc/http-template#
    >.
{
   ?photo exF:topic ?topic
}
=>
{
 _:request http:methodName "GET";
           http:requestURI ?topic;
           http:resp [http:body ?topic].
  ?photo foaf:primaryTopic ?topic
  ?topic a dbpedia:Book.
}.
```

Listing 1: GetTopic service

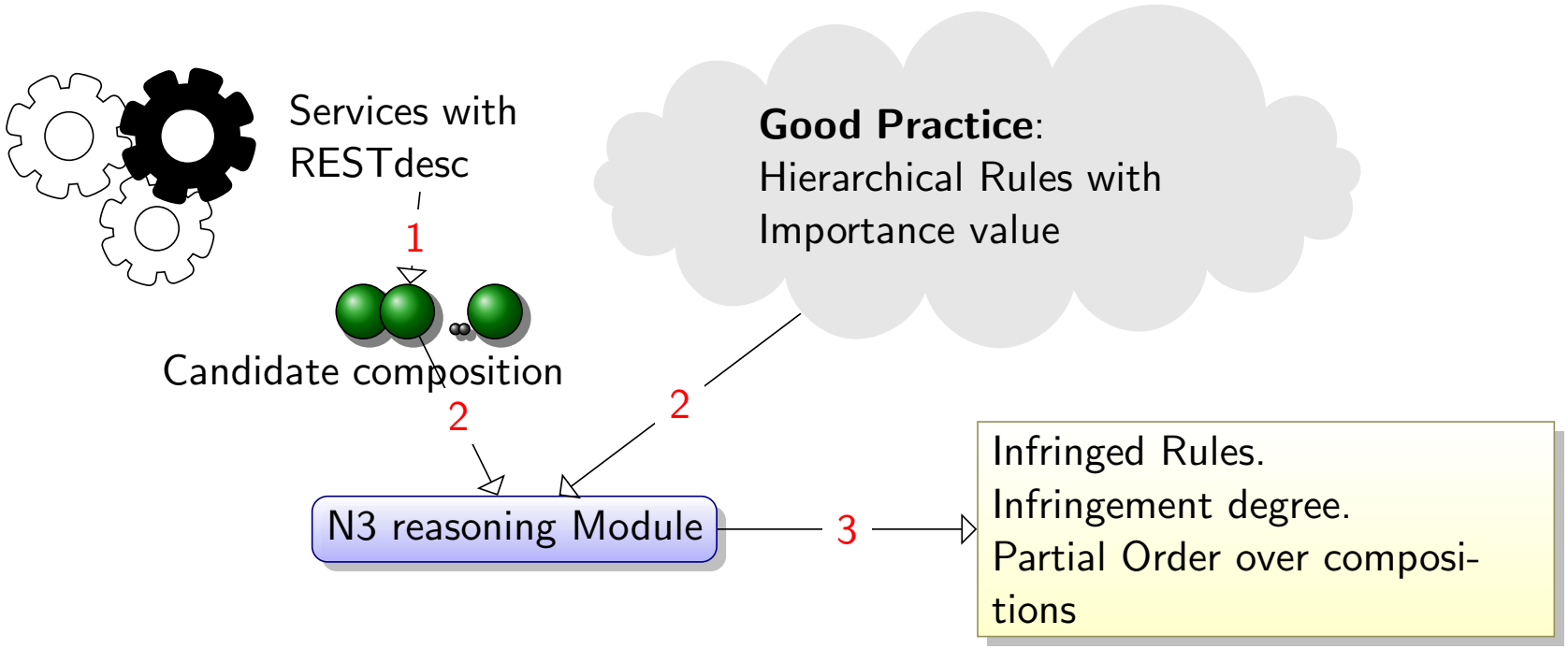
RESTdesc

```
{
   ?topic a dbpedia:Book .
   ?topic dbpedia2:name ?name
}
=>
{
 _:request http:methodName "GET";
           tmpl:requestURI ("/englishBook/" ?name);
           http:resp [http:body ?book].
  ?book dbpedia2:titleOrig ?name .
}.
```

Listing 2: GetEnglishVersion service

```
{ ?photo flicker:id ?id.
  ?photo foaf:primaryTopic _:work }
=>
{ _:work dbpedia:author ?author. }.
```

Domain-Specific Hierarchical Model



GoodPractice for Primitive Tasks

- $\langle name(o), precond(o), effects(o) \rangle$
- $effects(o) = \bigcup_i \langle cond(i), effect(i), value(i) \rangle$
- $value(i) \in V$ with partial order: $v_i \succ v_j$

1) *Task htn : hasPrecond ?Precond*
2) *Task htn : hasCons* r_i.
   3) r_i *htn : hasCond ?Cond.*
   4) r_i *htn : hasEffect*
      {_ : request http : resp ?Response.
      ConstraintsGraph$_i$*[?Response]}.*
   5) r_i *htn : hasValue* α_i.
   6) r_i *htn : hasComment TextualDescription.*

GoodPractice for Compound Tasks

- Method $m = \langle name(m),\ task(m),\ precond(m),\ network(m) \rangle$

$$CT \quad htn : isDescribedBy$$
$$\{[htn : hasContext \quad Context_i;$$
$$htn : hasDecomp(T_i^1 \quad T_2^i .... \quad T_i^i)]\}$$

Example: Good Practice for e-Commerce

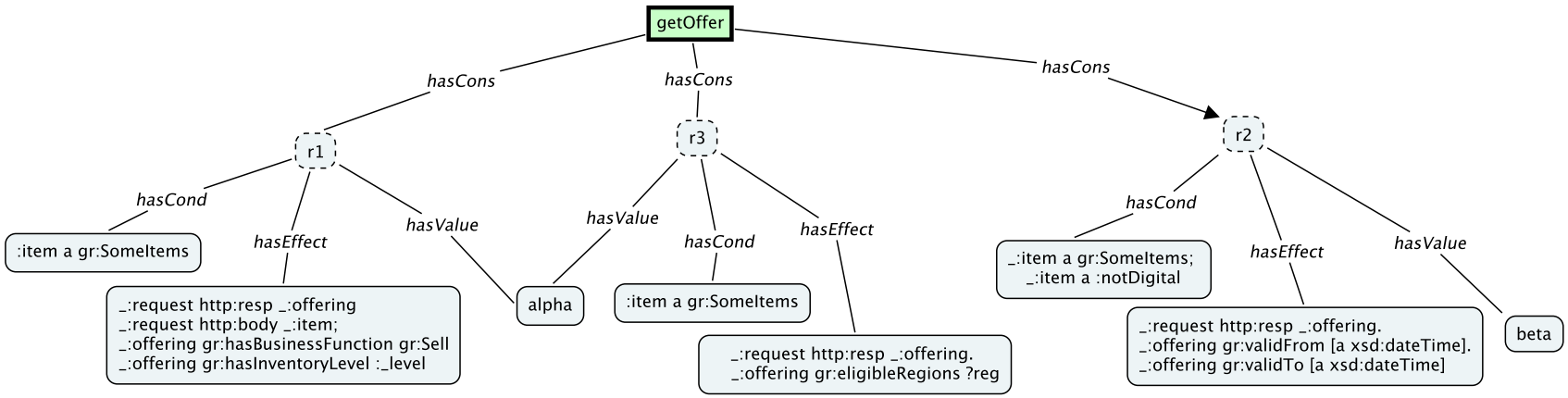


Figure: Description of GoodPractice for *getOffer* primitive task through three rules

Cont. Example - Primitive Task

```
@prefix gr: <http://purl.org/goodrelations/v1#> .
 :getOffer htn:hasConstraint :r1;
                htn:hasConstraint :r2;
                htn:hasConstraint :r3 .
:r1 htn:hasPrec {_:product a gr:SomeItems} .
:r1 htn:hasConc
   {_:request http:body _:product;
              http:resp _:offering.
    _:offering a gr:Offering.
    _:offering gr:includes _:product .
    _:offering gr:hasBusinessFunction gr:Sell .
    _:offering gr:hasInventoryLevel [a gr:
        QuantitativeValueFloat]}.
:r1 htn:hasComment "The response should include a
    selling offer for a product and it should indicate
     the stock quantity".
```

Cont. Example - Primitive Task

Other two rules and their value

```
:r2 :hasPrec {_:product a gr:SomeItems; _:product a :
    notDigital} .
:r2 htn:hasComment "The response should include offer
    's availability time.".

:r3 htn:hasComment "The response should include the
    eligible regions of an offering".

:r1  htn:hasImpDeg :alpha.
:r3  htn:hasImpDeg :alpha.
:r2  htn:hasImpDeg :beta.
:alpha > :beta.
```

Cont Example - Compound Task

Part of Good Practice for a buying Compound task

```
:buy htn:isDescribedBy
    [htn:hasContext {_:person a :RegisteredClient};
     htn:hasDecomp  (:search :authenticate :pay)].

:buy htn:isDescribedBy
 [htn:hasContext {_:person a :NewClient};
  htn:hasDecomp  (:search :setPersonalData :pay)].
```

Candidate Compositions Analysis: Representation

- RESTdesc compositions $\Rightarrow$ candidate composition C that includes $S_1, S_2, ..., S_n$
- Representation: C *htn* : *includes* S_i
- Semantics: the conjunction of the semantic of each service

```
{_:d e:findall (?Sem { ?Cand :includes ?S .
                       ?S log:semantics ?Sem}
                     ?List) .
 ?List log:conjunction ?SS} => {?Cand :hasSem ?SS}
     .
```

Operational semantic of primitive tasks

When a task is proposed for execution, its rules become active and the candidate composition that is checked must satisfy all these rules.

```
{ :state htn:do ?PrimitiveTask .
  ?PrimitiveTask htn:hasConstraint ?rule }
=>
  {:state htn:hasActive ?rule} .
```

Inference of infringement

Infringement of a rule is true iff antecedents are true but the consequences are false.

```
1 { ?C htn:hasSem ?SS.  :state htn:hasActive ?rule .
2  ?rule htn:hasPrec ?Prec. ?rule htn:hasConc ?Conseq.
3  ?SS log:supports ?Prec.  ?SS log:conclusion ?C.
4  ?C log:notIncludes ?Conseq }
5  => {?C htn:infringe ?rule}
```

Inference rule for choosing a decomposition

```
1. {:state htn:do ?M. ?M htn:isDescribedBy ?Descr .
2. ?Descr htn:hasContext ?Context.
3  ?Descr htn:hasDecomp ?Decomp.
4. :state htn:is ?C . ?C log:supports ?Context }
5.  => {:state htn:do ?Decomp}.
```

Execution of a decomposition

A recursive process for execution of all the included task

```
# the one element list
{:state htn:do ?L .  ?L rdf:first ?Act .
 ?L rdf:rest rdf:nil }
    => {:state htn:do ?Act} .
# the recursive access to the list
{:state htn:do ?L . ?L rdf:first ?Act .
 ?L rdf:rest ?R }
    => {:state htn:do ?Act . :state htn:do ?R} .
```

Compositions Ordering

- In possibilistic logic [**?**], a world ω is all the less possible as it falsifies formulas of higher degree.
- A candidate composition is less recommendable as it infringes a rule with higher importance degree
- Infringement degree *InfD* is the maximum degree of the rules from the set of infringed rules *InfR*

$$InfR(C) = \{r | (\mathcal{BP} \cup SS) \overset{\pi}{\models} \{C \quad htn : infringe \quad r\}\}$$

- A composition C_1 is preferred to composition C_2 stated as $C_1 \succ C_2$ iff $InfD(C1) < InfD(C2)$.
- A composition C is considered the most preferred composition iff it is safe and there is no other candidate composition better than it $\nexists C_i \ s.t. \ C_i \succ C$.

Example on ordering

- user goal involves the tasks *getOffer*.
- three candidates are found with: $InfR(C_1) = \{r1\}$, $InfR(C_2) = \{r2\}$, and $InfR(C_3) = \{r2, r3\}$
- *r1*: stock quantity, *r2*: availability time, *r3*: eligible regions
- r1, r2 *rightarrow alpha*, r3 *rightarrowbeta*, *alpha* $>$ *beta*
- $C_2 \succ C_1$ and also $C_2 \succ C_3$,

Conclusions

- a hierarchical model for Good Practice in service compositions
- a method in Notation N3 and N3Logic for using this knowledge in analysis of service compositions
  - identification of infringed rules
  - order relation over composition based on the infringed rules
- a hierarchical process that can easily be adapted for hierarchical task networks