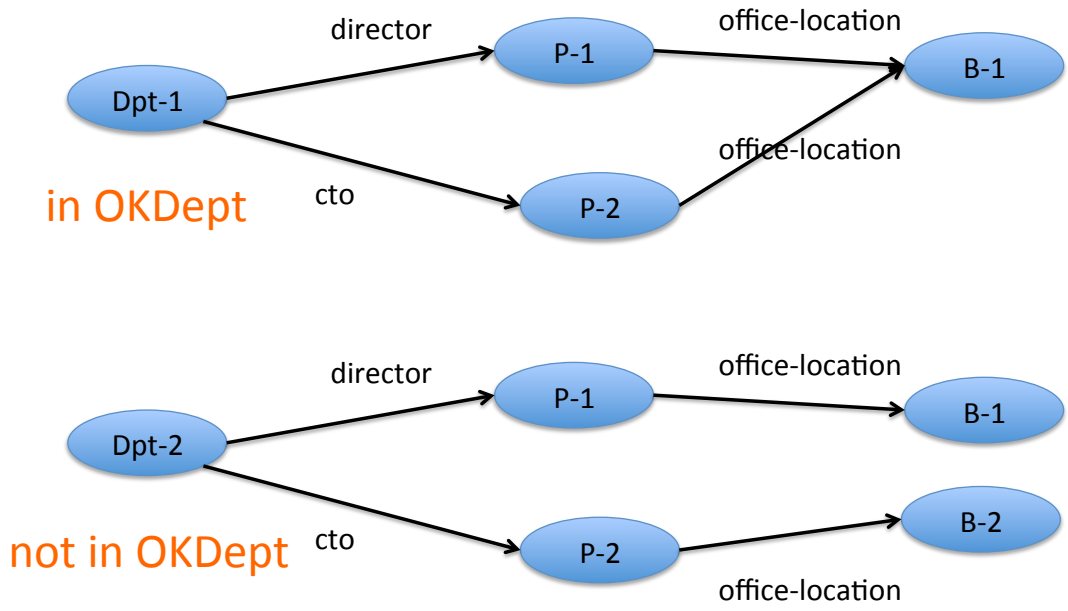
Infererence rules
SWRL

G. Falquet

Expressivity of DL

- DLs have the tree model property
  - +/- each set of axioms has a model that is a tree
  - Impossible to specify cyclic models

- Example
  - A department is in the class OKDept if and only if its director and chief technology officer are located in the same building



In DL (OWL 1)

Impossible to define OKDept

(possible in OWL-2 with Self)

Many other examples cannot be defined in OWL-2

Inference rules

Rules to produce

- New type assertions
  - x is a member of class C
- New property assertions
  - x is connected to Y through property P

SWRL Rules - syntax

rule ::= antecedant -> consequent

antecedant ::= atom, atom, ...
consequent ::= atom, atom, ...

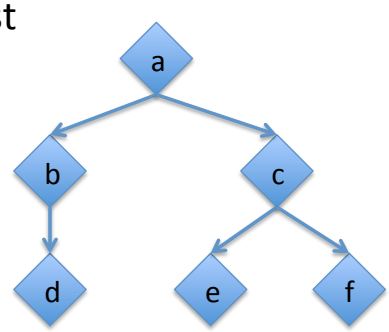
atom ::= description '(' i-object ')'
| dataRange '(' d-object ')'
| individualvaluedPropertyID '(' i-object i-object ')'
| datavaluedPropertyID '(' i-object d-object ')'
| sameAs '(' i-object i-object ')'
| differentFrom '(' i-object i-object ')'
| builtIn '(' builtinID { d-object } ')'

Person(?x), hasChild(?x, ?y), hasChild(?y, ?z) -> hasGrandChild(?x, ?z)

Interpretation

- Find all the variable bindings
  that satisfy the antecedent
- For each such binding the consequent must
  be satisfied

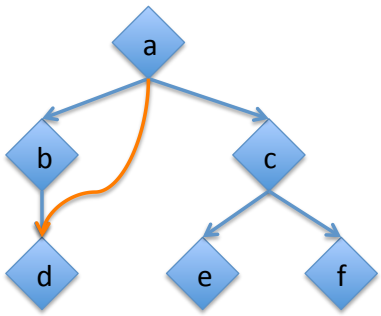
hasChild(?x, ?y), hasChild(?y, ?z)
-> hasGrandChild(?x, ?z)



Interpretation

hasChild(?x, ?y), hasChild(?y, ?z)
-> hasGrandChild(?x, ?z)

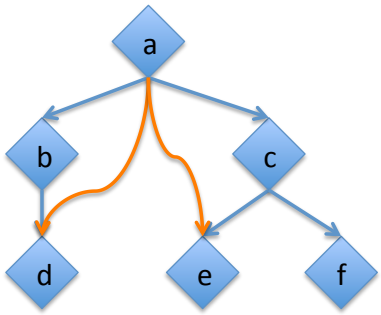
x=a, y=b, z=d -> hasGrandChild(a,d)



Interpretation

hasChild(?x, ?y), hasChild(?y, ?z)
-> hasGrandChild(?x, ?z)

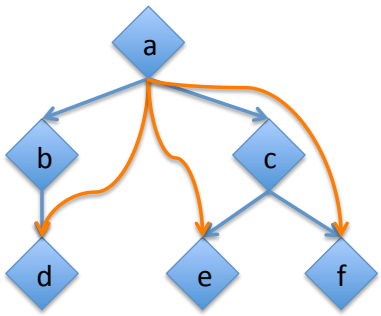
x=a, y=b, z=d -> hasGrandChild(a,d)
x=a, y=c, z=e -> hasGrandChild(a,d)



Interpretation

hasChild(?x, ?y), hasChild(?y, ?z)
-> hasGrandChild(?x, ?z)

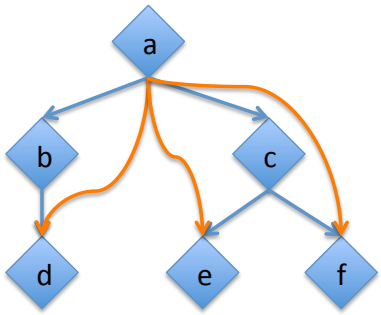
x=a, y=b, z=d -> hasGrandChild(a,d)
x=a, y=c, z=e -> hasGrandChild(a,d)
x=a, y=c, z=f -> hasGrandChild(a,d)



Interpretation

hasChild(?x, ?y), hasChild(?y, ?z)
-> hasGrandChild(?x, ?z)

x=a, y=b, z=d -> hasGrandChild(a,d)
x=a, y=c, z=e -> hasGrandChild(a,d)
x=a, y=c, z=f -> hasGrandChild(a,d)

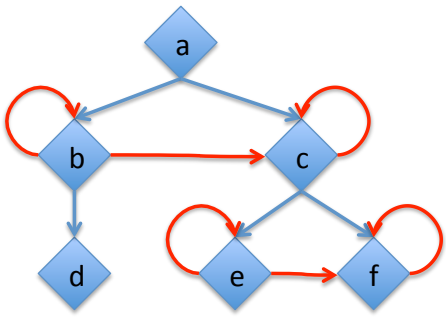


an interpretation that satisfies the rule

DifferentFrom

Variables with different names may represent the same individual !

hasChild(?x, ?y), hasChild(?x, ?z)
-> hasSibling(?y, ?z)

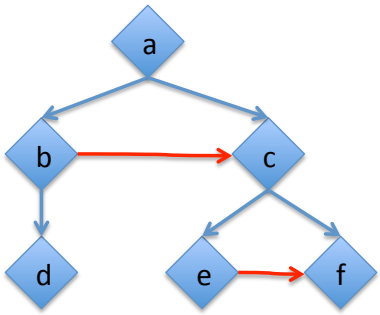


DifferentFrom

hasChild(?x, ?y), hasChild(?x, ?z), DifferentFrom (?y, ?z)
-> hasSibling(?y, ?z)

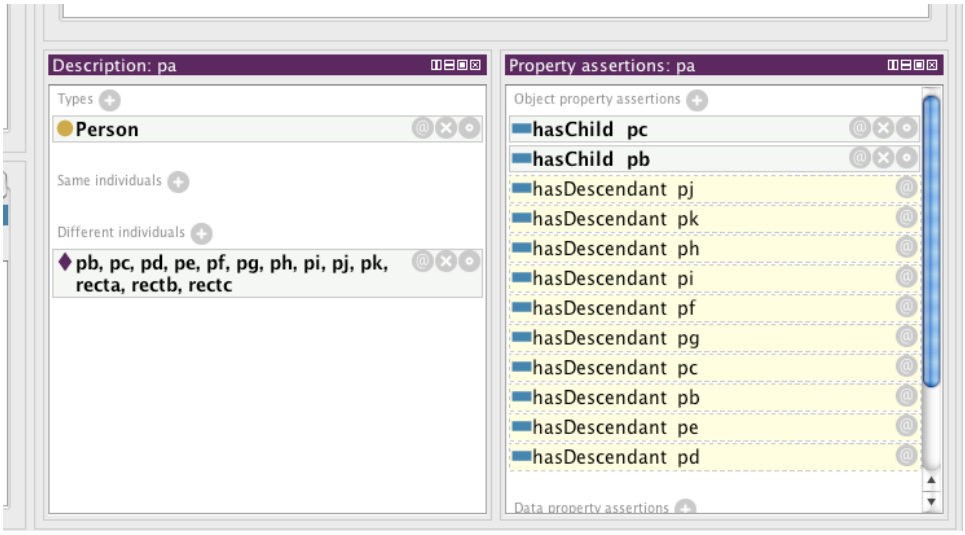
⚠ works only if

DifferentIndividual(b,c)
DifferentIndividual(e,f)



Example

hasChild(?x, ?y) -> hasDescendant(?x, ?y)
hasChild(?x, ?y), hasDescendant(?y, ?z) -> hasDescendant(?x, ?z)



DL-safe rules

Query answering for DL-axioms + rules is undecidable

It is decidable if rules are DL-safe

A rule r is called DL-safe if each variable in r occurs in a non-DL-atom in the rule body.

Practically: the variables in rules can only be bound to known individuals

Axioms:
Parent ≡ hasChild some Person
Parent(a), Parent(b), Parent(c),
Person(d),
hasChild(a,d)

Rule:
hasChild(?x, ?y) -> PersonWithChild(?x)

consequence:
PersonWithChild(a)

without the DL-safe restriction:
PersonWithChild(a), PersonWithChild(b), PersonWithChild(c)

Builtin predicates

To deal with numbers, strings, etc.

Rectangle(?x), hasWidthInMetres(?x, ?w), greaterThan(?w, 10)
-> WideRectangle(?x)

Rectangle(?x), hasHeightInMetres(?x, ?h),
hasWidthInMetres(?x, ?w),
greaterThan(?a, 100), multiply(?a, ?w, ?h)
-> LargeRectangle(?x)

swrlb:equal
swrlb:notEqual
swrlb:lessThan
swrlb:lessThanOrEqual
swrlb:greaterThan
swrlb:greaterThanOrEqual

swrlb:add
swrlb:subtract
swrlb:multiply
swrlb:divide
swrlb:integerDivide
swrlb:mod
swrlb:pow
swrlb:unaryPlus
swrlb:unaryMinus
swrlb:abs
swrlb:ceiling
swrlb:floor
swrlb:round
swrlb:roundHalfToEven
swrlb:sin
swrlb:cos
swrlb:tan

swrlb:stringEqualIgnoreCase
swrlb:stringConcat
swrlb:substring
swrlb:stringLength
swrlb:normalizeSpace
swrlb:upperCase
swrlb:lowerCase
swrlb:translate
swrlb:contains
swrlb:containsIgnoreCase
swrlb:startsWith
swrlb:endsWith
swrlb:substringBefore
swrlb:substringAfter
swrlb:matches
swrlb:replace
swrlb:tokenize

When you don't need SWRL: DL rules

Some SWRL rules can be encoded in OWL expressions

Example

Man(x) ⋀ hasBrother(x,y) ⋀ hasChild(y,z) → Uncle(x)

becomes

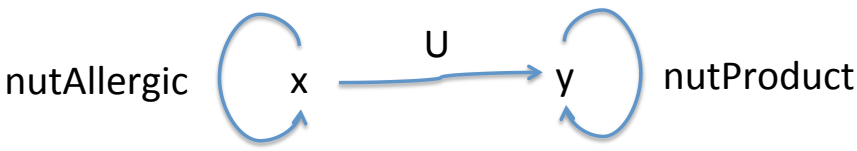
Man ⊓ ∃ hasBrother. ∃ hasChild.⊤ ⊑ Uncle

it's sometimes tricky ... (Hitzler & al. 2009)

NutAllergic(x) ⋀ NutProduct(y) → dislikes(x,y)

NutAllergic ≡ ∃ nutAllergic.Self
NutProduct ≡ ∃ nutProduct.Self
nutAllergic o U o nutProduct ⊑ dislikes

U = universal property (x U y is always true)

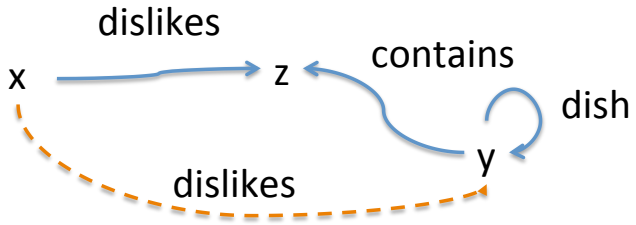


... more

dislikes(x,z) ⋀ Dish(y) ⋀ contains(y,z) → dislikes(x,y)

becomes

- **Dish ≡ ∃ dish.Self**
- **dislikes o contains⁻ o dish ⊑ dislikes**



Rules vs. SPARQL queries

- Rules are “executed” globally
  - all rules must be satisfied simultaneously
- Rules may have interactions
  - the outcome of a rule may trigger another one
- SPARQL queries are executed independently

Simulating rules with queries

define a 'construct' query for each rule

repeat

- execute each query
- add the results to the RDF graph

until nothing new is created

parent(?x, ?y) ⋀ ancestor(?y, ?z) → ancestor(?x, ?z)

```
construct {$x ancestor $z.}
where {$x parent $y. $y ancestor $z.}
```


SWRL and Protégé

There is a “rule” view in Protégé

to activate it:

- select the Entities tab
- menu Window -> Ontology Views -> Rules
- (a black dot appears)
- click the Class Annotation | Class Usage pane

The syntax uses “,” for the logical and (not “^”)

C(?x), p(?x, ?y) -> D(?y)

SameAs and DifferentFrom must start with an uppercase letter.

SWRL inference in Protégé

- Pellet and HermiT support SWRL inference
  - simply run the reasoner to perfom swrl inference
    - menu Reasoner -> Start Reasoner or Classify or Synchronize

- HermiT does not support the builtin atoms: add, multiply, lessThan, ... (=> hardly usable)

- Reasoners make the rules DL-safe by binding the variable only to explicitly asserted individuals
  - => reasoning is not complete with respect to the axioms

SWRL and Protégé bugs

Protégé 4.0

- the rule editor does not accept builtin predicates (add, multiply, lessThan, ...)

Protégé 4.1

- does not show the inferred data properties (the inferences actually takes place but the interface does not show them)

Consequence: you must work with both versions in parallel.

References

P. Hitzler, M. Krötzsch, S. Rudolph (2009) OWL 2 Rules. GeoS2009 Tutorial.

I. Horrocks et al. (2004) SWRL: A Semantic Web Rule Language Combining OWL and RuleML. W3C Memeber Submission. http://www.w3.org/Submission/SWRL/