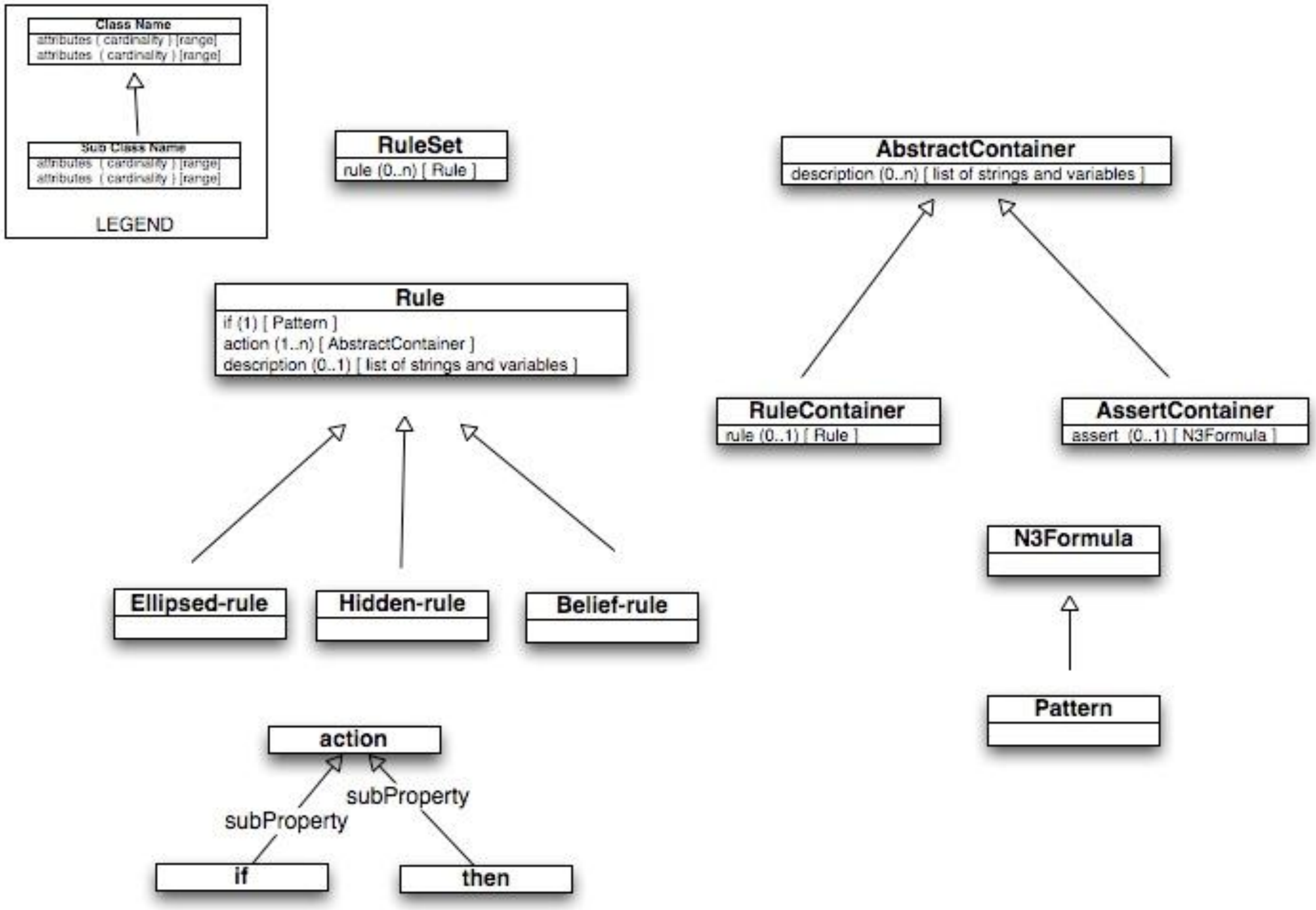
AIR Language : Ontology, Syntax & Semantics

03-01-2010

ankesh

AIR Ontology


Class Name
attributes (cardinality) [range]
attributes (cardinality) [range]
Sub Class Name
attributes (cardinality) [range]
attributes (cardinality) [range]
LEGEND
RuleSet
rule (0..n) [Rule]
AbstractContainer
description (0..n) [list of strings and variables]
Rule
if (1) [Pattern]
action (1..n) [AbstractContainer]
description (0..1) [list of strings and variables]
RuleContainer
rule (0..1) [Rule]
AssertContainer
assert (0..1) [N3Formula]
Ellipsed-rule
Hidden-rule
Belief-rule
N3Formula
Pattern
action
subProperty
subProperty
if
then

AIR Syntax (VD)

```
@forAll :Violation, :Work, :Creator, :ViolationDate, :DeathDate, :Value .

:CopyrightCriminalPolicy a air:Policy ;
             air:rule :FindInfringement .

:FindInfringement a air:Belief-rule ;
    air:if { @forSome :someVar ;
           :Violation a copyright:PotentialCopyrightInfringement ;
                  copyright:infringesCopyrightOn :Work . } ;
    air:then [ air:rule :FindValue ] .

:FindValue a air:Belief-rule ;
    air:if {
        :Work gr:hasCurrencyValue :Value ;
              gr:hasCurrency "USD" . } ;
    air:then [ air:rule :CheckValue ] .

:CheckValue a air:Belief-rule ;
    air:if {  :Value math:greaterThan "1000" . } ;
    air:else [ air:assert { :Violation air:non-compliant-with
                          :CopyrightCriminalPolicy . } ;
              air:description ( :Violation " is not a criminal copyright infringement as it is
    under $1,000 in value" ) ] ;
```

AIR Syntax (RS) 2

```
@forAll :Violation, :Work, :Creator, :ViolationDate, :DeathDate, :Value .

:CopyrightCriminalPolicy a air:Policy ;
            air:rule :FindInfringement .
            air:rule …;

:FindInfringement a air:Belief-rule ;
    air:if {
           :Violation a copyright:PotentialCopyrightInfringement ;
                    copyright:infringesCopyrightOn :Work . } ;
    air:then [ air:rule :FindValue ] .

:FindValue a air:Belief-rule ;
    air:if {
        :Work gr:hasCurrencyValue :Value ;
              gr:hasCurrency "USD" . } ;
    air:then [ air:rule :CheckValue ] .

:CheckValue a air:Belief-rule ;
    air:if {  :Value math:greaterThan "1000" . } ;
    air:else [ air:assert { :Violation air:non-compliant-with
                      :CopyrightCriminalPolicy . } ;
              air:description ( :Violation " is not a criminal copyright infringement as it is
    under $1,000 in value" ) ] ;
```

AIR Syntax (R)

```
@forAll :Violation, :Work, :Creator, :ViolationDate, :DeathDate, :Value .

:CopyrightCriminalPolicy a air:Policy ;
             air:rule :FindInfringement .
             air:rule :SomeOtherRule;

:FindInfringement a air:Belief-rule ;
    air:if {
           :Violation a copyright:PotentialCopyrightInfringement ;
                    copyright:infringesCopyrightOn :Work . } ;
    air:then [ air:rule :FindValue ] ;
    air:then [ ….] …

:FindValue a air:Belief-rule ;
    air:if {
        :Work gr:hasCurrencyValue :Value ;
              gr:hasCurrency "USD" . } ;
    air:then [ air:rule :CheckValue ] .

:CheckValue a air:Belief-rule ;
    air:if {  :Value math:greaterThan "1000" . } ;
    air:else [ air:assert [ { :Violation air:non-compliant-with
                          :CopyrightCriminalPolicy . } ;
               air:description ( :Violation " is not a criminal copyright infringement as it is
    under $1,000 in value" ) ] ;
```

New

OLD

```
@forall :person, :father, :brother.
If { :person hasfather :father .}
then [ if {:father hasbrother :brother }
                then [ air:assert {:person hasUncle :brother.} ];
                 else [air:assert { :person hasUncle false .} ].
```

changed

```
@forall :person, :father.
If { :person hasfather :father .}
then [ if {@forSome :brother.
                          :father hasbrother :brother }
                ~~then [ air:assert {:person hasUncle :brother.} ];~~ # define new rule
                 else [ air:assert { :person hasUncle false .} ].
```

AIR Syntax (AC)

```
@forAll :Violation, :Work, :Creator, :ViolationDate, :DeathDate, :Value .

:CopyrightCriminalPolicy a air:Policy ;
             air:rule :FindInfringement .
             air:rule :SomeOtherRule;

:FindInfringement a air:Belief-rule ;
    air:if {
           :Violation a copyright:PotentialCopyrightInfringement ;
                    copyright:infringesCopyrightOn :Work . } ;
    air:then [ air:rule :FindValue ] .

:FindValue a air:Belief-rule ;
    air:if {
        :Work gr:hasCurrencyValue :Value ;
              gr:hasCurrency "USD" . } ;
    air:then [ air:rule :CheckValue ] .

:CheckValue a air:Belief-rule ;
    air:if {  :Value math:greaterThan "1000" . } ;
    air:else [ air:assert { :Violation air:non-compliant-with
                          :CopyrightCriminalPolicy . } ;
               air:description ( :Violation " is not a criminal copyright infringement as it is
    under $1,000 in value" ) ] ;
```

AIR Syntax (des) 5

```
@forAll :Violation, :Work, :Creator, :ViolationDate, :DeathDate, :Value .

:CopyrightCriminalPolicy a air:Policy ;
             air:rule :FindInfringement .
             air:rule :SomeOtherRule;

:FindInfringement a air:Belief-rule ;
    air:if {
           :Violation a copyright:PotentialCopyrightInfringement ;
                    copyright:infringesCopyrightOn :Work . } ;
    air:then [ air:rule :FindValue ] .

:FindValue a air:Belief-rule ;
    air:if {
        :Work gr:hasCurrencyValue :Value ;
              gr:hasCurrency "USD" . } ;
    air:then [ air:rule :CheckValue ] .

:CheckValue a air:Belief-rule ;
    air:if {  :Value math:greaterThan "1000" . } ;
    air:else [ air:assert { :Violation air:non-compliant-with
                          :CopyrightCriminalPolicy . } ;
              air:description ( :Violation " is not a criminal copyright infringement as it is
    under $1,000 in value" ) ] ;
```

AIR Syntax (B)

```
@forAll :Violation, :Work, :Creator, :ViolationDate, :DeathDate, :Value .

:CopyrightCriminalPolicy a air:Policy ;
             air:rule :FindInfringement .
             air:rule :SomeOtherRule;

:FindInfringement a air:Belief-rule ;
    air:if {
           :Violation a copyright:PotentialCopyrightInfringement ;
                    copyright:infringesCopyrightOn :Work . } ;
    air:then [ air:rule :FindValue ] .

:FindValue a air:Belief-rule ;
    air:if {
        :Work gr:hasCurrencyValue :Value ;
              gr:hasCurrency "USD" . } ;
    air:then [ air:rule :CheckValue ] .

:CheckValue a air:Belief-rule ;
    air:if {  :Value math:greaterThan "1000" . } ;
    air:else [ air:assert { :Violation air:non-compliant-with
                      :CopyrightCriminalPolicy . } ;
           air:description ( :Violation " is not a criminal copyright infringement as it is
    under $1,000 in value" ) ] ;
```

Cwm built-ins:
- •Maths
- •time
- •String
- •List
- •Xml
- •Log
- •Sparql
- •…
- •air:justifies

Example using AIR justifies

```
:CheckApplicableCopyright a air:Beliefrule;
    air:if { @forSome :log, :rule .
            <./data.n3> log:semantics :log .
            <./copyright.n3> log:semantics :rule.
          ( ( :log ) ( :rule ) ) air:justifies { <./data.n3#Work>
    air:compliantwith :CopyrightPolicy . } } .
    air:then [ air:assert [ air:statement {:Violation
    air:compliantwith :CriminalInfringementPolicy . } ] ] .
```

AIR Syntax 6

```
@forAll :Violation, :Work, :Creator, :ViolationDate, :DeathDate, :Value .

:CopyrightCriminalPolicy a air:Policy ;
             air:rule :FindInfringement .

:FindInfringement a air:Belief-rule ;
    air:if {
           :Violation a copyright:PotentialCopyrightInfringement ;
                    copyright:infringesCopyrightOn :Work . } ;
    air:then [ air:rule :FindValue ] .

:FindValue a air:Belief-rule ;
    air:if {
        :Work gr:hasCurrencyValue :Value ;
              gr:hasCurrency "USD" . } ;
    air:then [ air:rule :CheckValue ] .

:CheckValue a air:Belief-rule ;
    air:if {  :Value math:greaterThan "1000" . } ;
    air:else [ air:assert { :Violation air:non-compliant-with
                          :CopyrightCriminalPolicy . } ;
               air:description ( :Violation " is not a criminal copyright infringement as it is
    under $1,000 in value" ) ] ;
```

AIR Syntax

```
@forAll :Violation, :Work, :Creator, :ViolationDate, :DeathDate, :Value .

:CopyrightCriminalPolicy a air:Policy ;
             air:rule :FindInfringement .
             air:rule :SomeOtherRule;

:FindInfringement a air:Belief-rule ;
    air:if {
           :Violation a copyright:PotentialCopyrightInfringement ;
                    copyright:infringesCopyrightOn :Work . } ;
    air:then [ air:rule :FindValue ] .

:FindValue a air:Ellipse-rule ; #similarly air:Hidden-rule
    air:if {
        :Work gr:hasCurrencyValue :Value ;
              gr:hasCurrency "USD" . } ;
    air:then [ air:rule :CheckValue ] .

:CheckValue a air:Belief-rule ;
    air:if {  :Value math:greaterThan "1000" . } ;
    air:else [ air:assert { :Violation air:non-compliant-with
                          :CopyrightCriminalPolicy . } ;
              air:description ( :Violation " is not a criminal copyright infringement as it is
    under $1,000 in value" ) ] ;
```

Justification - Example Log 1

```
:MinorInfringement a
  copyright:PotentialCopyrightInfringement ;
    copyright:infringesCopyrightOn :SpaceOdyssey ;
    copyright:infringementDate "2008-10-01" .

:SpaceOdyssey a movie ;
    gr:hasCurrencyValue "30" ;
    gr:hasCurrency "USD" ;
    dc:creator :StanleyKubrick .
```

URI : <http://dig.csail.mit.edu/TAMI/inprogress/example-log.n3>

```
* - this triple is considered part of the premise only for
  this example.
```

Justification (Belief-rule) 2

```
pr:Event4 a airj:RuleApplication;
   air:rule :CheckValue;
   pmll:outputdata { :MinorInfringement air:non-compliant-with
   :CopyrightCriminalPolicy . };
   air:description ":MinorInfringement is not a criminal copyright infringement as
   it is under $1,000 in value";
   airj:branch true;
   airj:dataDependency pr:GetDataLog ;
   airj:flowDependency pr:Event3 .

pr:Event3 a airj:RuleApplication;
   air:rule :FindValue;
   airj:outputVariableMappingList (pr:Mapping3, pr:Mapping2, pr:Mapping1);
   air:branch true ;
   airj:dataDependency pr:GetDataLog ;
   airj:flowDependency pr:Event2 .

pr:Mapping3 a pmlj:Mapping;
    pmlj:mappingFrom :Value;
    pmlj:mappingTo   “30”.
```

Justification – Ellipsed Rule 3

```
pr:Event4 a airj:RuleApplication;
    air:rule :CheckValue;
    pmll:outputdata { :MinorInfringement air:non-compliant-with
    :CopyrightCriminalPolicy . };
    air:description ":MinorInfringement is not a criminal copyright infringement as
    it is under $1,000 in value";
    airj:branch true;
    airj:dataDependency pr:GetDataLog ;
    airj:flowDependency pr:Event3 .

@forSome pr:Event3 .
pr:Event3 a airj:RuleApplication;
    air:rule :FindValue;
    airj:outputVariableMappingList (pr:Mapping3, pr:Mapping2, pr:Mapping1);
    air:branch true ;
    airj:dataDependency pr:GetDataLog ;
    airj:flowDependency pr:Event2 .

pr:Mapping3 a pmlj:Mapping;
     pmlj:mappingFrom :Value;
     pmlj:mappingTo   "30".
```

Justification – Hidden Rule 4

```
@forSome pr:Event4 .
pr:Event4 a airj:RuleApplication;
    air:rule :CheckValue;
    pmll:outputdata { :MinorInfringement air:non-compliant-with
    :CopyrightCriminalPolicy . };
    air:description ":MinorInfringement is not a criminal copyright infringement as
    it is under $1,000 in value";
    airj:branch true;
    airj:dataDependency pr:GetDataLog ;
    airj:flowDependency pr:Event3 .

pr:Event3 a airj:RuleApplication;
    air:rule :FindValue;
    airj:outputVariableMappingList (pr:Mapping3, pr:Mapping2, pr:Mapping1);
    air:branch true ;
    airj:dataDependency pr:GetDataLog ;
    airj:flowDependency pr:Event2 .

pr:Mapping3 a pmlj:Mapping;
     pmlj:mappingFrom :Value;
     pmlj:mappingTo   "30".
```

Semantics 1

Top rule, nesting, then/else actions

```
@forAll :Violation, :Work, :Creator, :ViolationDate, :DeathDate .

:CopyrightCriminalPolicy a air:Policy ;
             air:rule :FindInfringement .
             air:rule :SomeOtherRule;

:FindInfringement a air:Belief-rule ;
    air:if {
           :Violation a copyright:PotentialCopyrightInfringement ;
                    copyright:infringesCopyrightOn :Work . } ;
    air:then [ air:rule :FindValue ] .

:FindValue a air:Belief-rule ;
    air:if {
        :Work gr:hasCurrencyValue :Value ;
              gr:hasCurrency "USD" . } ;
    air:then [ air:rule :CheckValue ] .

:CheckValue a air:Belief-rule ;
    air:if {  :Value math:greaterThan "1000" . } ;
    air:else [ air:assert { :Violation air:non-compliant-with
                          :CopyrightCriminalPolicy . } ;
              air:description ( :Violation " is not a criminal copyright infringement as it is
    under $1,000 in value" ) ] ;
```

Semantics 2

0. Only top rules are active initially
1. Add top rules to active rules set (ARS).
2. Satisfy Rules in ARS. Execute ‘then’ actions of rules whose conditions are satisfied. (fire a new rule and/or assert a triple to the KB)
3. repeat 2 until no more ‘then’ actions can be taken
4. Choose the first* failed rule, execute it’s ‘else’ action.
5. Repeat 2 .. 5.

*queue – nested rules are pushed in later

Semantics (Algorithm) 2

0. Only top rules are active initially
1. Add top rules to active rules set (ARS).
2. Satisfy Rules in ARS. Execute ‘then’ actions of rules whose conditions are satisfied.
    1. add rule to ARS
    2. assert a triple to the KB
3. repeat 2 until no more ‘then’ actions can be taken
4. Choose the first* failed rule, execute its ‘else’ action.
    1. add rule to ARS
    2. assert a triple to the KB
5. Repeat 2 .. 5.

*queue – nested rules are pushed in later

Thanks