Rule-Based Trust Assessment on the Semantic Web

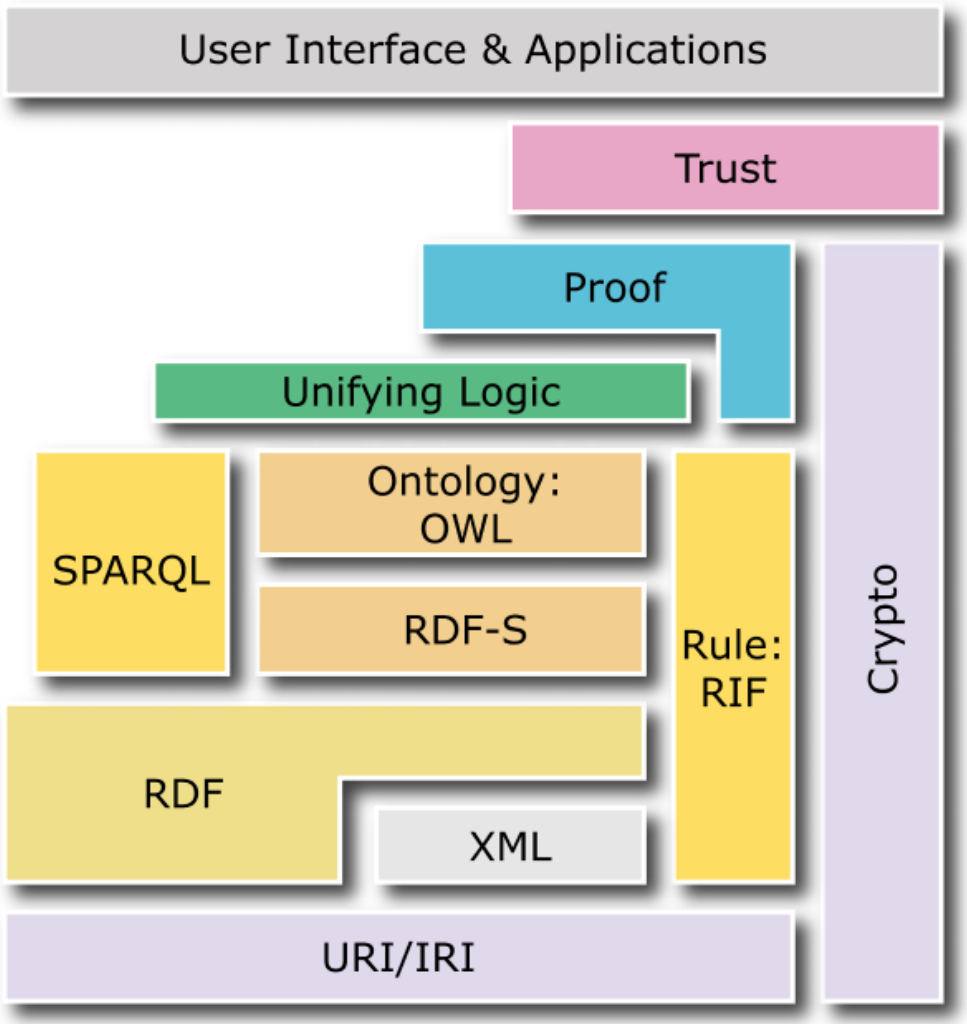
Ian Jacobi[1], Lalana Kagal[1], **Ankesh Khandelwal**[2]

1 *Decentralized Information Group*
Massachussets Institute of Technology

2 *Tetherless World Constellation*
Rensselaer Polytechnic Institute

07/21/2011

Semantic Web Layer-cake



Tim-Berners Lee 2005

Annotation of rules

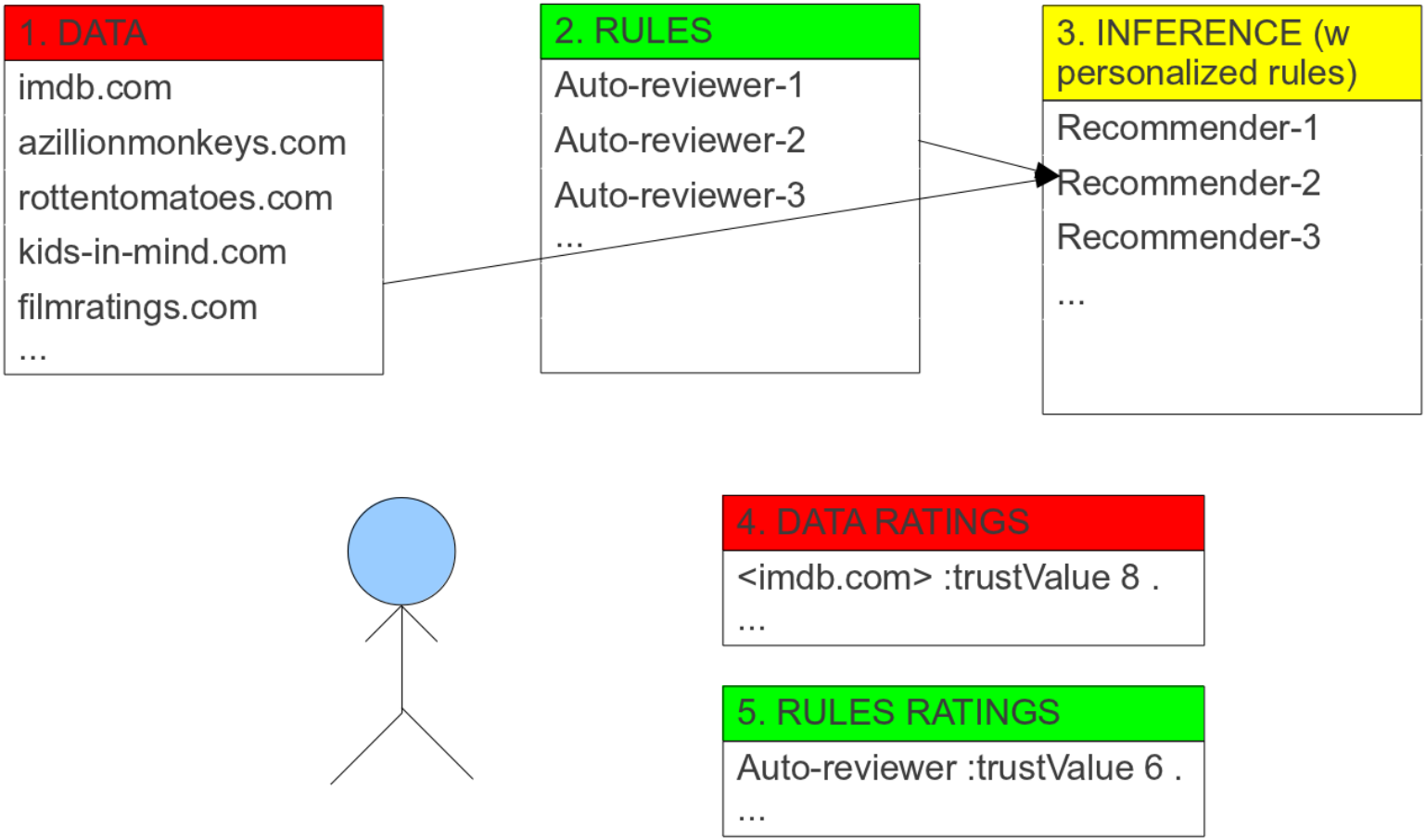
- Rules will be reused more on the Web as domains acquire widely-used ontologies
- E.g. rule-based policies, inter-organizational business rules, policies and practices, medical decision support
- Reasons for variable trust
  - Variable domain knowledge
  - Variable expertise in writing rules
  - Short-hand for quick computations, possibly complete but unsound inferences.
  - Non-explicit assumptions
  - Malicious intents

Outline

General model

- **Trust**: belief, confidence, recentness etc.
- **Trust categories**: content-based and meta-data-based [Bizer 96]
- **Trust axes**: data and rules
- **Trust computation model**: formal algebraic structure [Straccia 10] or mixed trust representations and their flexible combination

Example scenario: movie recommendations



Outline

AIR semantic web (production) rules language

Accountability **I**n **R**DF

- N3-based; graphs used as literal values
- Rules are resources
- Rule description:

```
:ruleid if {graph-pattern}; then <actions>;
else <actions> .
:actionid rule :ruleid .
:actionid assert {graph-pattern} .
```

- Compatible with N3-Logic and Cwm built-ins
- Justification ontology in N3

Trust representation in N3

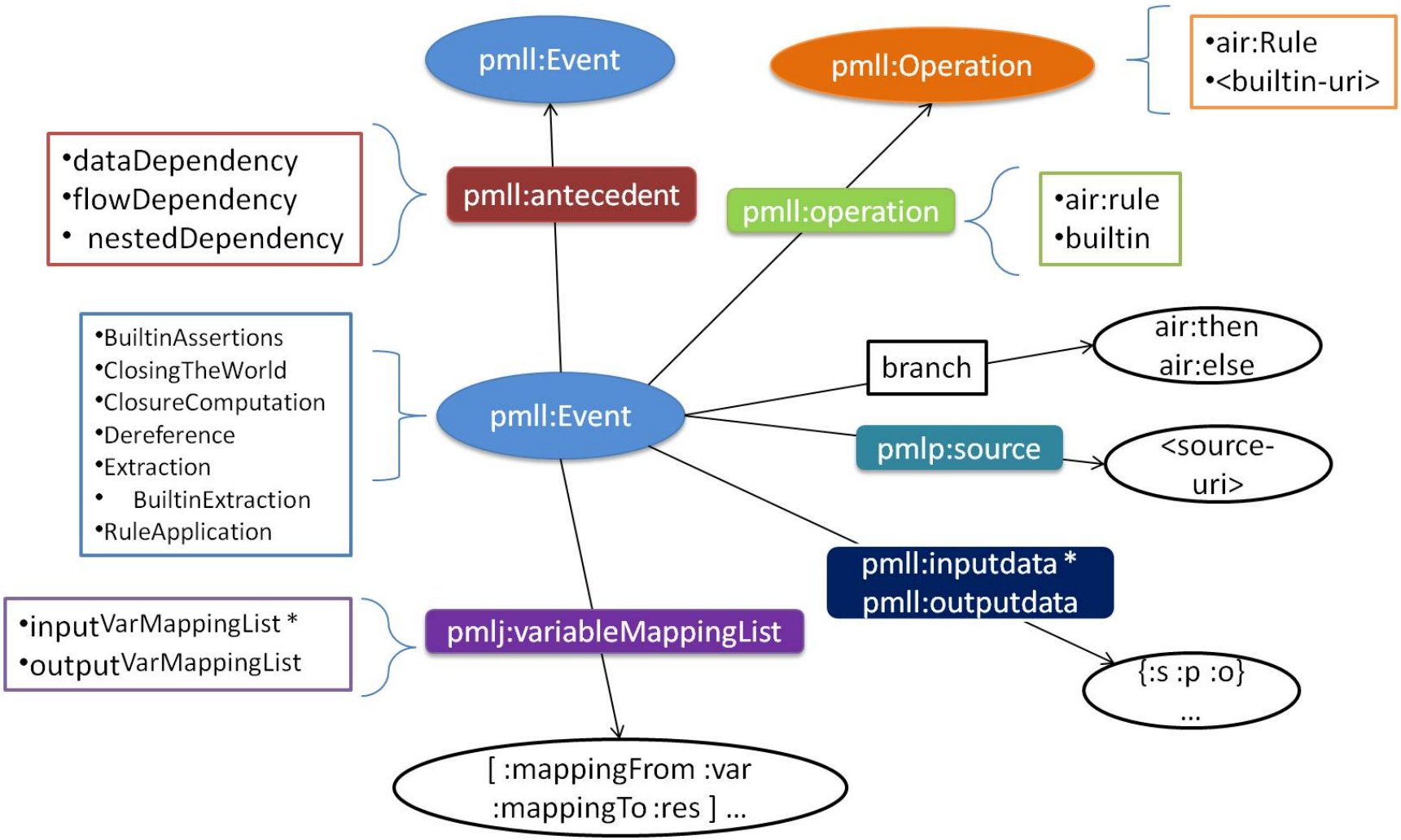
- `:Recommender-1 rdf:type :Trusted .`
- `:Mary :trustsHighly :Recommender-2 .`
- `<http://www.imdb.com> :trustValue 0.7 .`
- `{:Mary :canWatch :HP} :trustValue 0.8 .`

Example of AIR rule

1. `:ruleid-1 a air:Belief-rule`
2. ```
   :ruleid-1 air:if {
         ?reco says {:Mary :canWatch ?movie} .
         :Mary :trustsHighly ?reco
   }
   ```
3. `:ruleid-1 air:then _:b`
4. ```
   _:b air:assert {
         :Mary believes {:Mary :canWatch ?movie}
   }
   ```

Mary believes recommendation for a movie only when recommended by someone she trusts highly.

AIR justification ontology



Justification triples

1. `:ruleapp a air:RuleApplication .`
2. `:ruleapp pmll:outputdata {inferred-triples} .`
3. `:ruleapp pmll:operation :ruleid .`

4. `:ruleapp pmll:dataDependency :extract .`
5. `:extract a air:Extraction .`
6. `:extract pmll:source <source-uri> .`

Outline

Trust on rules

- may be assigned separately from the rule definitions
- Different entities may have different trust on same rules
- may not be uniform for all rules in a rule-base
- Justifications or proofs may be used to compute trust on inferred triples
- E.g. `:auto-reviewer :trustValue 0.7 .`

Example: trust on inferred triples by trust on rules

1. `:ruleid-3 a air:Belief-rule`
2. ```
   :ruleid-3 air:if {
         ?reco says {:Mary :canWatch ?movie} .
         ?app pmlj:outputdata {:Mary :canWatch ?movie}.
         ?app pmll:operation ?ruleid .
         ?ruleid :tVal ?tRule
   }
   ```
3. `:ruleid-3 air:then _:b`
4. ```
   _:b air:assert {
         {:Mary :canWatch ?movie} :tVal ?tRule
   }
   ```

Mary trusts recommendations for a movie to the degree that it trusts the general rule (auto-reviewer) used to come to that conclusion.

Example: trust on inferred triples by trust on rules and input data

1. `:ruleid-4 a air:Belief-rule`
2. `:ruleid-4 air:if {`
   `?reco says {:Mary :canWatch ?movie} .`
   `?app pmlj:outputdata {:Mary :canWatch ?movie}.`
   `?app pmll:operation ?ruleid .`
   `?ruleid :tVal ?tRule .`
   `?app pmll:dataDependency ?extract.`
   `?extract pmll:source ?d-source .`
   `?d-source :tVal ?tData .`
   `(?tRule ?tData) math:product ?tComb`
   `}`
3. `:ruleid-4 air:then _:b`
4. `_:b air:assert {`
   `{:Mary :canWatch ?movie} :tVal ?tComb`
   `}`

Outline

Related work

- Computing trust for explicitly asserted RDF data *[Richardson 03, Gil 02, Golbeck 03]*
- WIQA framework *[Bizer 06]*
- SAOR *[Hogan 08]*
- Reasoning with annotated semantic web data- calculating trust on inferred triples *[Straccia 10]*

Summary and future work

- Resources on the Web including **rules** are subject to trust
- Trust on any rule-based inference is a function of trust on both data and rules (used for inference)
  - No formal work for reasoning with annotated rules
- Trust on inferred statements can be computed from proofs
- N3 equally suitable for representing trust on RDF resources and statements
- AIR rule language can be used for flexible trust assessment of inferred statements.
- Methodologies for finer trust assignments, and trust assessments for rules

Acknowledgements

We thank **Jim Hendler**, **Gregory Williams**, **Maryam Fazel-Zarandi** (U. Toronto) and **Jiao Tao** for their feedback on this presentation.

Resource Description Framework

Figure: **An RDF Graph describing Eric Miller** [http://www.w3.org/TR/rdf-syntax/]

- <http://www.w3.org/People/EM/contact#me> rdf:type <http://www.w3.org/2000/10/swap/pim/contact#Person>
- **rdf** for ***http://www.w3.org/TR/rdf-syntax#***

Graph identification

- Named graphs: multiple RDF graphs, named with URIs, in a single document or repository.[1]
- N3: extends RDF; graph as literals.
- Next version of RDF; SPARQL already supports it.
- (RDF-reification is not very helpful.)

[1] J. J. Carroll et al. *Named graphs, provenance and trust*. In WWW '05.

Example: Trust on inferred triples in negative rules

1. `:ruleid-x a air:Belief-rule`
2. `:ruleid-x air:if {`
   `{?res a ?cls} :tVal ?tType.`
   `{?cls rdfs:subClassOf ?super} :tVal ?tSco.`
   `(?tType ?tSco) math:product ?tComb.`
   `?tComb math:greaterThan 0.7`
   `}`
3. `:ruleid-x air:else _:b`
4. `_:b air:assert {`
   `{?res a ?super} :tVal 0`
   `}`

- *Assumption*. ?res & ?super are bound. ?cls is existentially quantified along with ?tType, ?tSco, & ?tComb.
- If type cannot be inferred from any `rdfs:subClassOf` axiom with trust more than 0.7 then trust on that type is 0.

Selective trust on patterns of information

- Finer trust association- different trust for different information from same source.
- E.g. Hospital may be trusted with information about potential virus outbreak but not for economic predictions.

```
:source :isTrustedWith _:b .
_:b rdf:type :TrustInfo .
_:b :tPattern { pattern } .
_:b :tValue trust-val .
```

- Similar to WIQA policies; in addition can associate degree of trust.
- Trust values assigned to triples separate from data; same triple may be trusted differently in various documents.

Interpretation of selective trust association

1. `:ruleid a air:Belief-rule`
2. ```
   :ruleid air:if {
           :source log:semantics ?graph .
            ?graph log:includes {pattern}
   }
   ```
3. `:ruleid air:then _:b`
4. ```
   _:b air:assert {
           {pattern} :tVal trust-val
   }
   ```

- **`log:semantics`**, built-in for getting N3 graph from N3/RDF document
- **`log:includes`**, built-in for graph inclusion.
- `:tVal` is short-hand for `:trustVal`
- *Note.* variable symbols are quantified URIs; in the presentation we use literal strings that start with '?' instead.

Example: Trust on inferred triples

with greater than threshold trust

1. `:ruleid-3 a air:Belief-rule`
2. ```
   :ruleid-3 air:if {
         {?res a ?cls} :tVal ?tType.
         {?cls rdfs:subClassOf ?super} :tVal ?tSco.
         (?tType ?tSco) math:product ?tComb.
         ?tComb math:greaterThan 0.7
   }
   ```
3. `:ruleid-3 air:then _:b`
4. ```
   _:b air:assert {
         {?res a ?super} :tVal ?tComb
   }
   ```

- Trust on `rdf:type` inference through a subClassOf axiom is product of trusts in type information of resource and the subClassOf axiom, when product is more than 0.7, else 0.

Some considerations

- Trust management (TM) is usually convenient when handled transparently.
- TM explicitly within rules to deal with diversity in trust metrics, value-types or representations,
- and for using multiple trust calculi in different rules.
- TM scenarios covered here can be simulated by any reasoner that can manage annotations of RDF triples and has built-ins for simple trust value computations.
  - Further, rules may be referenceable and annotatable.
  - One of the examples required that justifications for inference (or proof) also be manipulable through rules.
  - Aggregate built-ins such as `max` useful in *generalization*, when inferred triples have multiple trust values.

Example of AIR rule

Subclass-based type inference

1. `:ruleid-1 a air:Belief-rule`
2. ```
   :ruleid-1 air:if {
         ?res a ?cls .
         ?cls rdfs:subClassOf ?super
   }
   ```
3. `:ruleid-1 air:then _:b`
4. ```
   _:b air:assert {
         ?res a ?super
   }
   ```

Example of AIR rule with annotated triples

Subclass-based type inference

1. `:ruleid-2 a air:Belief-rule`
2. ```
   :ruleid-2 air:if {
         {?res a ?cls} :tVal ?tType.
         {?cls rdfs:subClassOf ?super} :tVal ?tSco.
         (?tType ?tSco) math:product ?tComb
   }
   ```
3. `:ruleid-2 air:then _:b`
4. ```
   _:b air:assert {
         {?res a ?super} :tVal ?tComb
   }
   ```

- Trust on `rdf:type` inference through a subClassOf axiom is product (or any other *t-norm*) of trusts in type information of resource and the subClassOf axiom.

Example: Trust on inferred triples by trust on rules

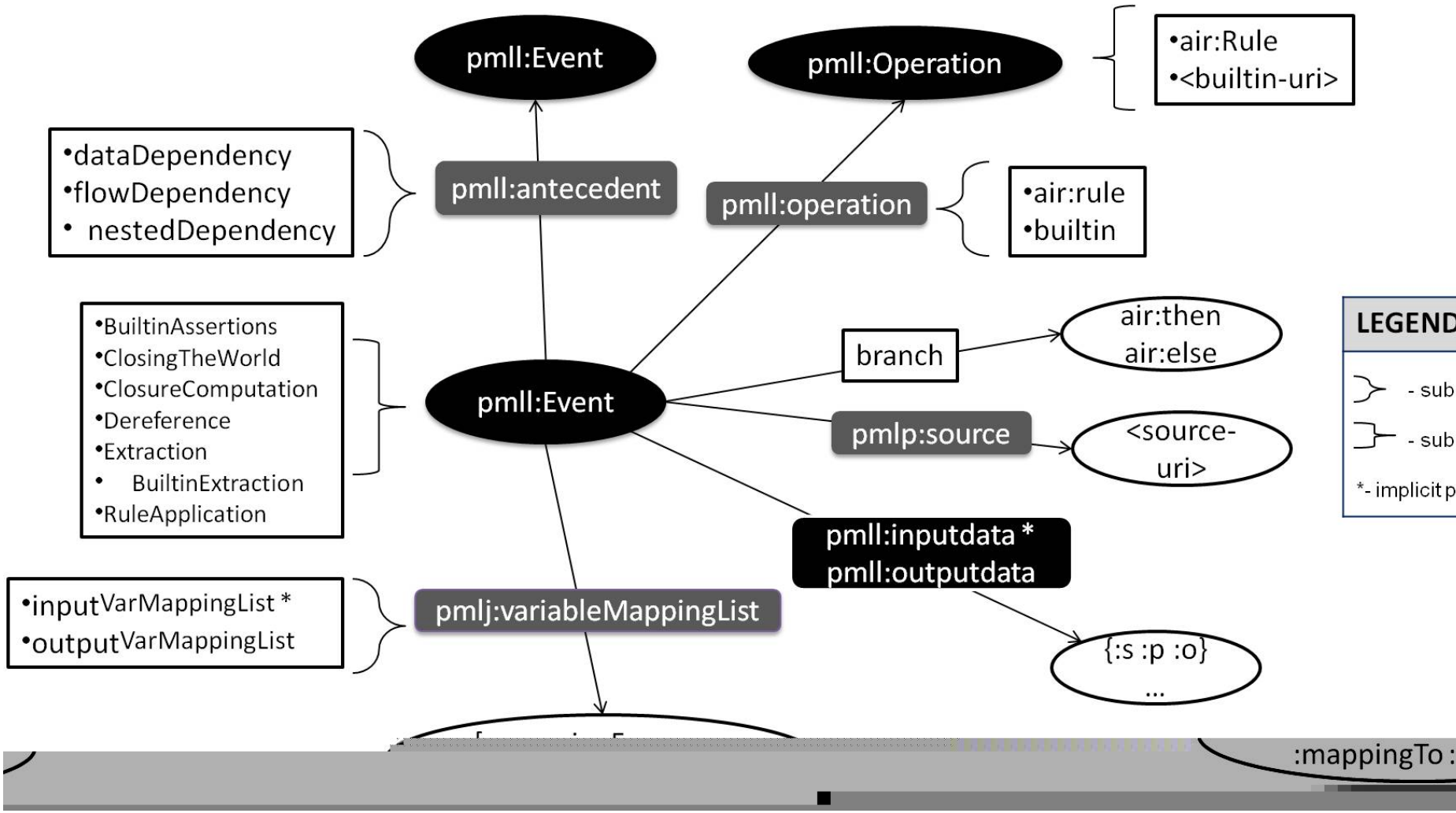
1. `:ruleid-3 a air:Belief-rule`
2. ```
   :ruleid-3 air:if {
         ?s ?p ?o .
         ?ruleapp pmlj:outputdata {?s ?p ?o} .
         ?ruleapp pmll:operation ?ruleid .
         ?ruleid :tVal ?tRule .
   }
   ```
3. `:ruleid-3 air:then _:b`
4. ```
   _:b air:assert {
         {?s ?p ?o} :tVal ?tRule .
   }
   ```

Given justification for inferred triple, trust on it is modified by trust on rule involved in its inference (w/o taking dependencies into account).

Example: Trust on inferred triples by trust on rules and input data

```
1 :ruleid-4 a air:Belief-rule
2 :ruleid-4 air:if {
        ?s ?p ?o .
        ?ruleapp pmlj:outputdata {?s ?p ?o} .
        ?ruleapp pmll:operation ?ruleid .
        ?ruleid :tVal ?tRule .
        ?ruleapp pmll:dataDependency ?extract.
        ?extract pmll:source ?d-source .
        ?d-source :tVal ?tData .
        (?tRule ?tData) math:product ?tComb
  }
3 :ruleid-4 air:then _:b
4 _:b air:assert {
        {?s ?p ?o} :tVal ?tComb .
  }
```

AIR justification ontology



Example: Provenance-based trust assignment via rules

```
:BadSensorRule a air:Belief-rule .

@forAll :DATA, :PROCESS, :TIME, :UNIXTIME .

IF    :DATA a opm:Artifact ;
           opm:wasGeneratedBy :PROCESS ;
           opm:wasGeneratedAt :TIME .
      :PROCESS owl:sameAs :BadSensorProcess .
      :TIME time:inSeconds :UNIXTIME .

THEN  activate-rule :BadSensorTimeRule
```

```
:BadSensorTimeRule a air:Belief-rule .

IF    :UNIXTIME math:greaterThan :BadSensorFixedTime .

THEN  assert :DATA t:trustvalue 7.

ELSE  assert :DATA t:trustvalue 3.
```

Fig. 7. Deriving trust from OPM provenance: `BadSensorRule` uses provenance metadata about some datum encoded in OPM to assign trust to the datum depending on the time the data was generated by the faulty sensor.