Policy Assurance for PIR Queries

Lalana Kagal
MIT CSAIL
Decentralized Information Group

Overview

- Project Introduction
  - Problem statement
  - Policy assurance architecture
  - Challenges
- Previous results
  - AIR policy language & reasoner
  - Justification UI

Overview

- Current results:
  - Use case development
  - N3 representation for SPARQL
  - Demo compliance/non-compliance for simple queries & policies
- Next steps
  - Develop methodology to convert abstract policies to more data-specific compliance rules
  - Policy editor
  - Include support for SQL queries either converting SQL to RDF directly or via SPARQL

Problem Statement

- Policy assurance - Proving that queries made by the client conform to mandated policies preventing leakage of unauthorized information
  - What kind of language should be used to express these policies ?
  - What tools and techniques will help encourage rule-following and identify non-compliance ?

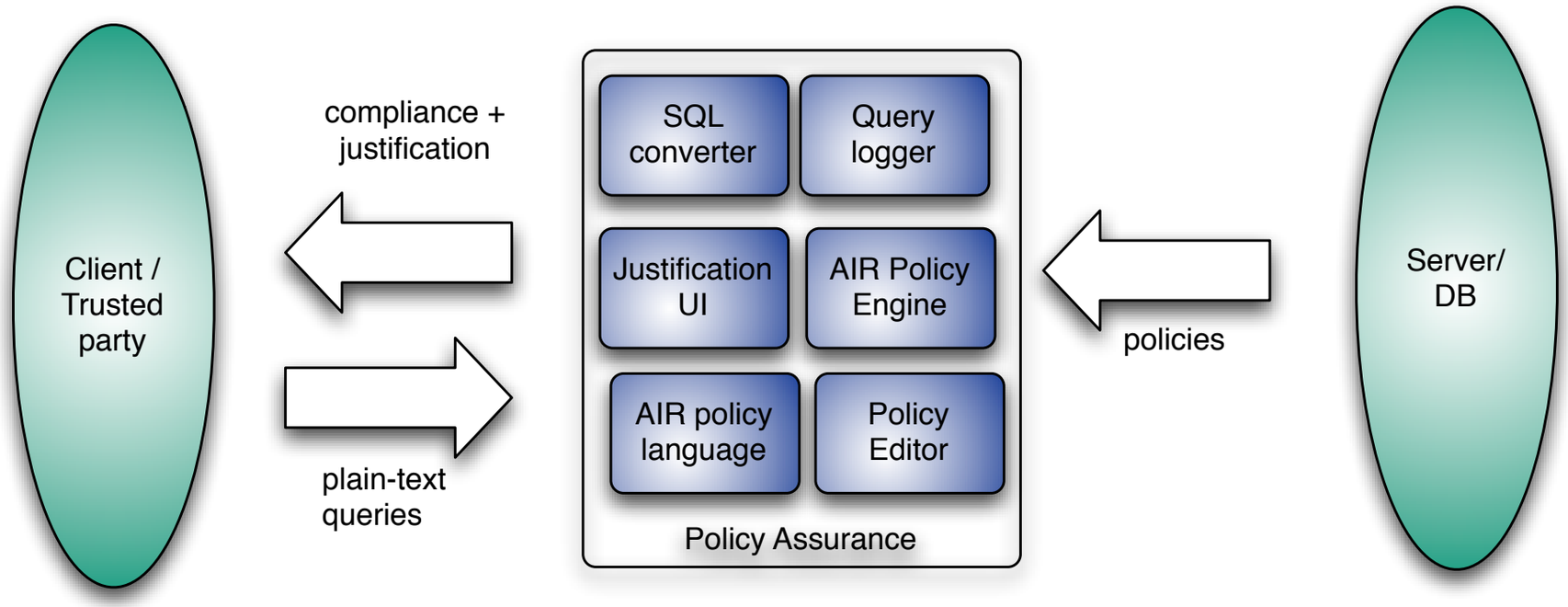
Challenges in Policy Assurance

- Not enough to restrict certain keywords or rows or cols; policies are ambiguous
  - For example, “Access to SSN is not permitted”.
  - Does this mean that SSN values cannot be retrieved or does it also include use of SSN values to filter the results
  - Notions of strong versus weak compliance
- Policies are usually rule-based
  - For example, “Access to marital status, gender, and religion for US citizens is not permitted”
- Policies tend to deal in abstract terms and talk about kinds of information that should not be accessible or should not be used for certain purposes
  - For example, “Access to contact information for minors is not permitted”, or “my health information cannot be used to contact me regarding experimental drugs”
- Need a specification language that is able to capture the semantics of query compliance policies

Challenges in Policy Assurance

- Though individual queries might not violate privacy policies, a certain combination of queries might lead to a violation.
  - For example, an analyst might generate a query that yields a target's alias and then query a different database that establishes the true identity associated with that pseudonym. If that real identity reveals that the target is a US Person, then certain subsequent queries would violate various laws and executive orders, even though a simple analysis of the query itself would not reveal a violation.
- Just identifying non-compliance of a query not enough
  - does not help the client understand why the query failed
  - require reason or justification for non-compliance

Assurance Architecture



Policy Assurance Components

- Query logger
- AIR Policy language
  - a machine-understandable policy language for expressing privacy policies
  - Semantic Web technologies for shared model of data
- AIR Reasoning engine
  - for reasoning over queries and policies to identify violations
  - justifications
  - Handle private policies

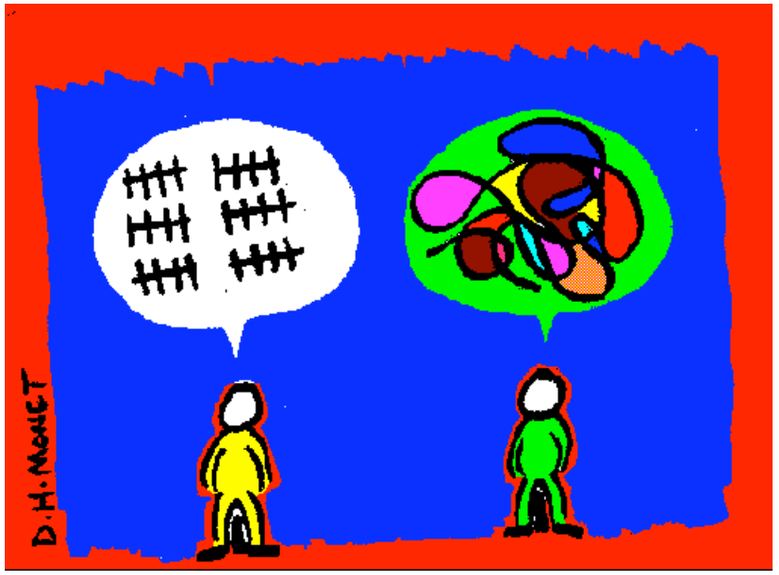


Image courtesy of http://home.ca.inter.net/~dmonet/

Policy Assurance Components

- Justification User Interface
  - Why UI ?
  - Graphical justification interface that will provide a structured natural language explanation for policy non-compliance
- SQL converter
  - Convert SQL into format understandable by AIR reasoner
- Policy development
  - Support definition of high level policies de-coupled from query and database structure

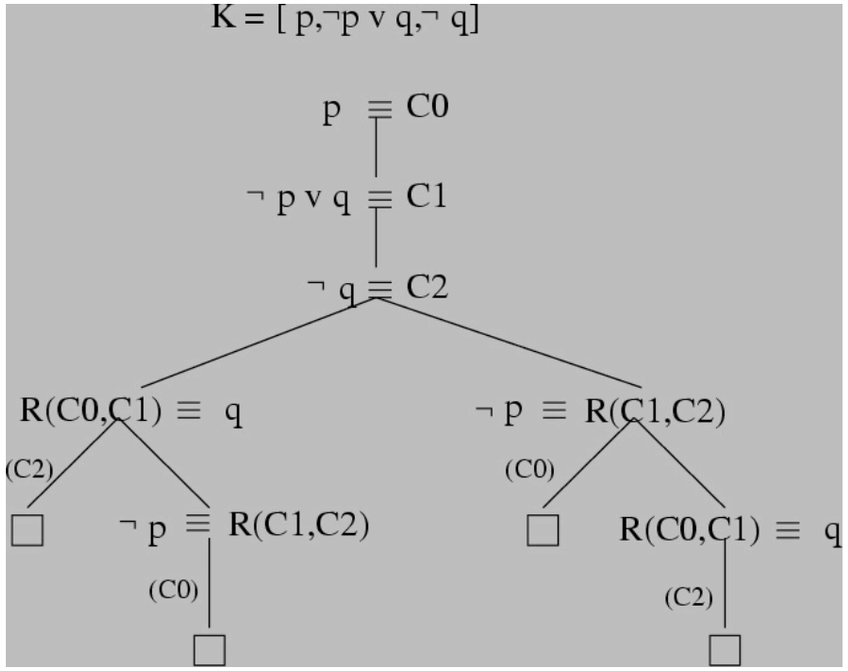


Image courtesy http://clip.dia.fi.upm.es/~logalg/slides/

Previous research

- AIR policy language based on Semantic Web technologies
  - shared model of discourse
  - global unique identifiers
  - interoperability - mapping between terms possible via properties such as subClassOf, SameAs, equivalentProperty, etc.
    - Example - mit:Student subClassOf foaf:Person

Previous research

- AIR Policy reasoner able to identify compliant and non-compliant AIR policies
  - production-rule system that features pattern matching, dependency tracking, and nesting of rules
  - Generates a justification for each conclusion
- Tabulator Firefox Extension

```
@prefix : <http://dig.csail.mit.edu/data#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix air: <http://dig.csail.mit.edu/TAMI/2007/amord/air#> .
@prefix tms: <http://dig.csail.mit.edu/TAMI/2007/amord/tms#> .
@prefix yosi: <http://dig.csail.mit.edu/People/yosi#> .

   :DAP_1     tms:justification tms:premise .

   :DAP_3     tms:description  (
      :Req2
     " is a request made by a requester with openid, "
     <http://auth.mit.edu/syosi>
     ", for DIG resource "
     <http://dig.csail.mit.edu/proposals/nsf.tex/> );
      tms:justification  [
         tms:antecedent-expr  [
            a tms:And-justification;
            tms:sub-expr :DAP_1,
                 {:DIG     :owns
                    <http://dig.csail.mit.edu/proposals/nsf.tex/> .
                  :Req2     a air:Request;
                       air:resource
                       <http://dig.csail.mit.edu/proposals/nsf.tex/>
;
                   foaf:openid <http://auth.mit.edu/syosi> .
                  } ];
         tms:rule-name :DAP_1 ] .

   :Req2     air:compliant-with :DIGPolicy .
```

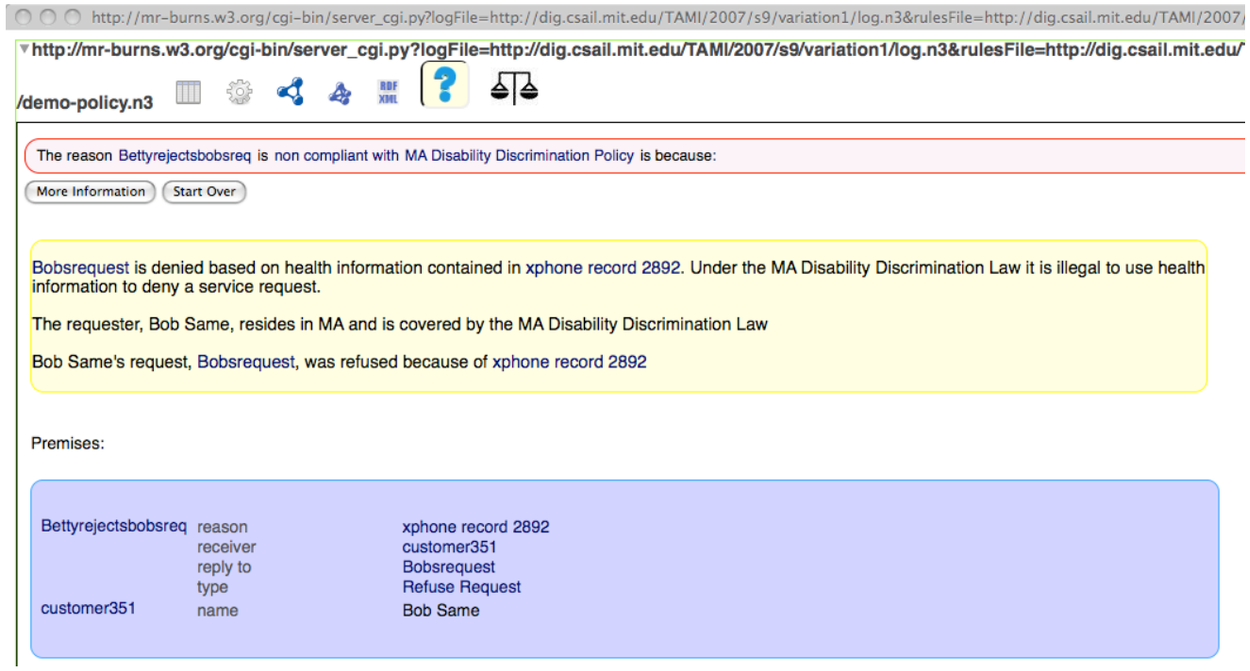
1

```
   {
      :Req2     air:compliant-with :DIGPolicy .
   }
   tms:description  (
      "The requester with openid, " <http://auth.mit.edu/syosi>
      ", is known to a DIG member, "
      <http://dig.csail.mit.edu/People/RRS> );
    tms:justification  [
       tms:antecedent-expr  [
          a tms:And-justification;
           tms:sub-expr :DAP_3,
           {<http://dig.csail.mit.edu/People/RRS>    air:in
                 :MemberList;
                 foaf:knows yosi:YES .
            yosi:YES foaf:openid  <http://auth.mit.edu/syosi> .
           } ];
        tms:rule-name :DAP_3 ] .

   {
     <http://dig.csail.mit.edu/People/RRS>     air:in
:MemberList;
     foaf:knows yosi:YES .
     yosi:YES    foaf:openid  <http://auth.mit.edu/syosi> .
     :DIG     :owns <http://dig.csail.mit.edu/proposals/nsf.tex/> .
     :Req2     a air:Request;
        air:resource <http://dig.csail.mit.edu/proposals/nsf.tex/>;
        foaf:openid <http://auth.mit.edu/syosi> .
     }   tms:justification tms:premise .
```

2

Proof tree generated by AIR reasoner



Justification UI in Tabulator

Current research

- Use case 0 development
  - Policy: SSN numbers cannot be accessed or used to filter queries
  - 6 example queries
    - 4 non-compliant
    - 2 compliant
- N3 serialization for SPARQL

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?s ?id ?n WHERE {
  ?s foaf:age ?a.
  ?s foaf:openid ?id.
  OPTIONAL { ?s foaf:ssn ?n }.
  FILTER ( ?a > 18 )
}
```

SPARQL query

```
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix math: <http://www.w3.org/2000/10/swap/math#>.
@prefix s: <http://dig.csail.mit.edu/2009/IARPA-PIR/sparql#> .
@prefix : <http://dig.csail.mit.edu/2009/IARPA-PIR/query1#> .

:Query-5 a s:Select;
   s:cardinality :ALL;
   s:POSList [
      s:variable :S;
      s:variable :N;
      s:variable :ID;
    ];
   s:WhereClause  :WHERE.

   :WHERE a s:DefaultGraphPattern;
       s:TriplePattern  { :S <http://xmlns.com/foaf/0.1/age> :A };
       s:TriplePattern  { :S <http://xmlns.com/foaf/0.1/openid> :ID };
       s:Filter [
          a s:ComparatorExpression;
          s:TriplePattern { :A s:BooleanGT "18"^^xsd:integer }
        ];
       s:OptionalGraphPattern [
             s:TriplePattern  { :S <http://xmlns.com/foaf/0.1/ssn> :N };
        ].

#ends
```

SPARQL query in N3

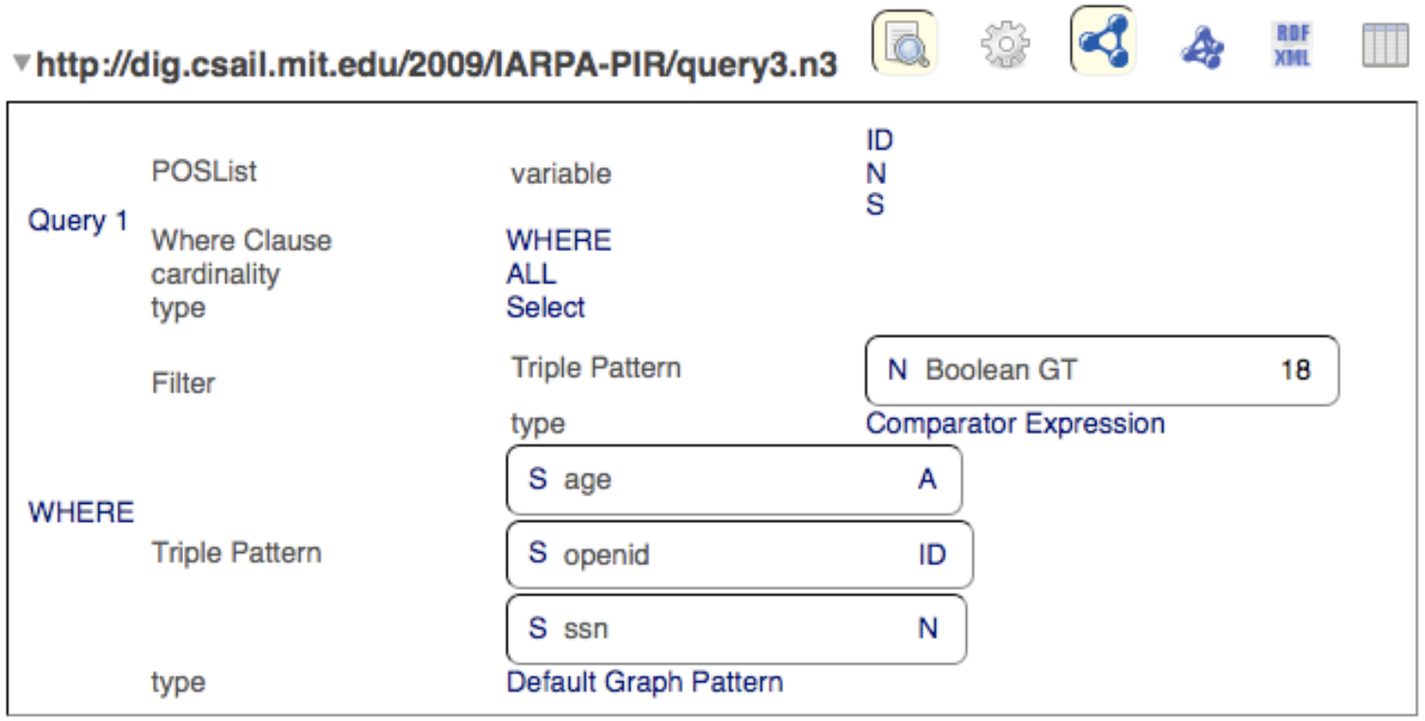
Current research

- Demo compliance/non-compliance for simple queries & policies

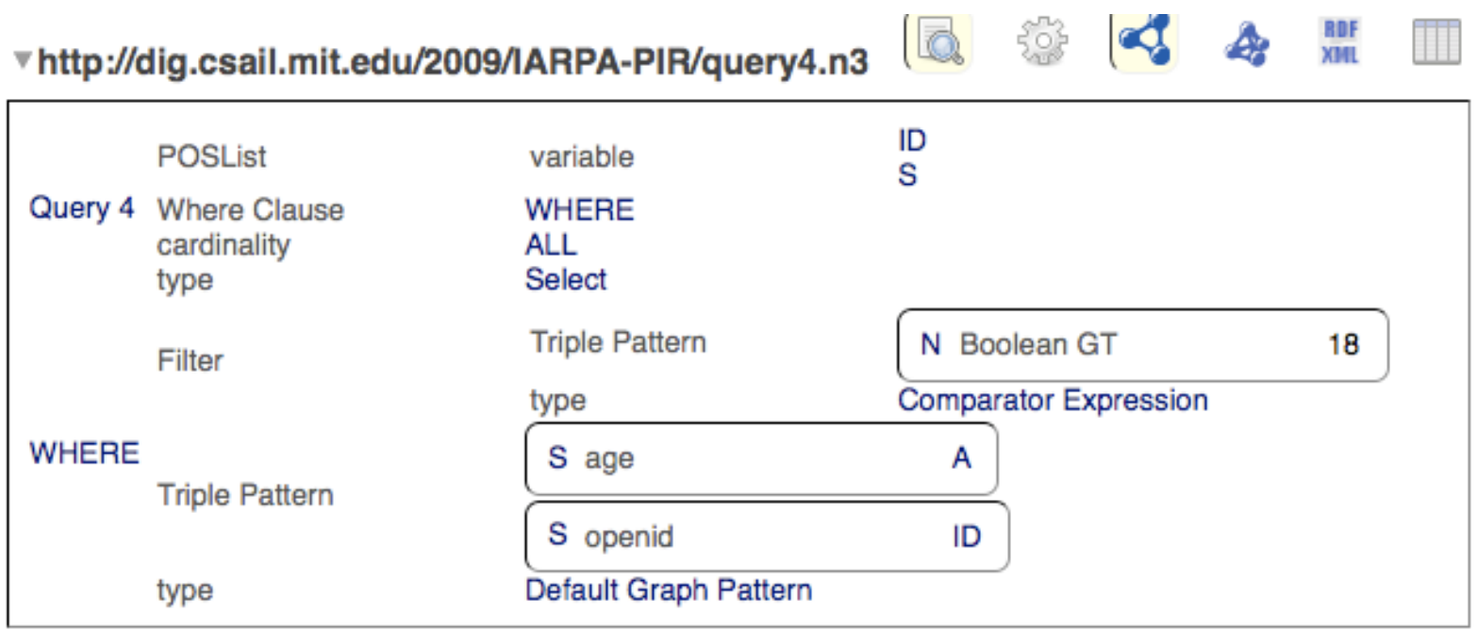
SSNPolicy	label	SSN sample policy for IARPA PIR project
	rule	SSN RULE1
	type	Policy
	description	Q is a SPARQL query
SSN RULE1	label	SSN policy rule1
	pattern	Q POSList P Q Where Clause W Q type Select
	rule	SSN RULE2 SSN RULE3 SSN RULE4 SSN RULE5
	type	Belief Rule
	assert	Q non compliant with SSNPolicy
	description	The query, Q , uses SSN values in the where clause, as a filter and retrieves SSN values as well
SSN RULE2	label	SSN policy rule2
	pattern	P variable V W Filter F W Triple Pattern T T includes X ssn V F Triple Pattern S S includes V @@: _:n228
	type	Belief Rule
	assert	Q non compliant with SSNPolicy
	description	The query, Q , uses SSN values in the where clause and retrieves SSN values
SSN RULE3	label	SSN policy rule3
	pattern	P variable V W Triple Pattern T

Part of SSN policy

Current research

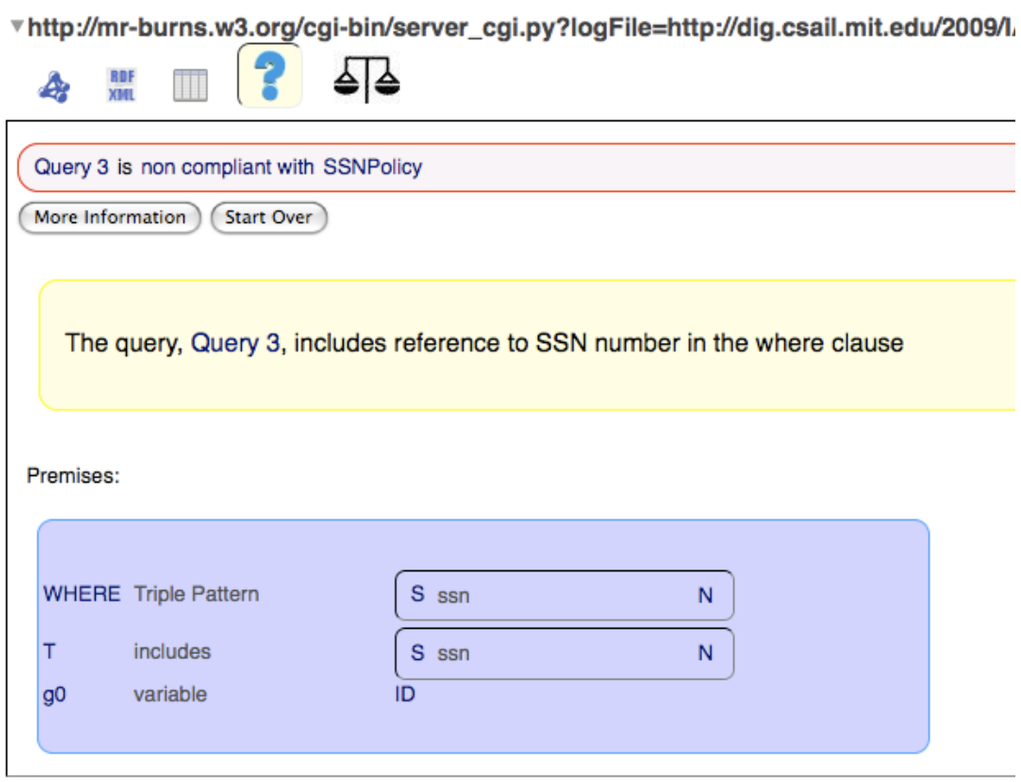


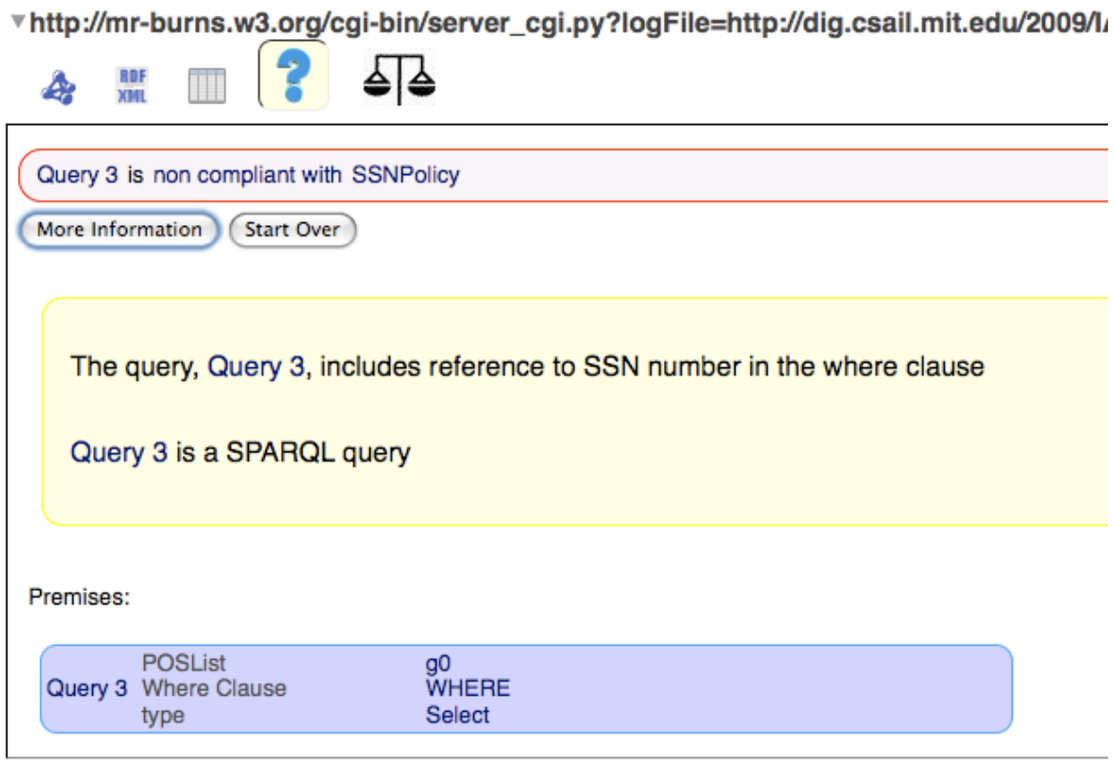
Non-compliant query



Compliant query

Current research





Next steps

- Automate conversion of queries
  - Extend SPASQL so that we can automatically convert SPARQL queries to N3
  - Include support for SQL queries either converting SQL to N3 directly or via SPARQL
- Convert sets of queries and policies prepared by the test and evaluation team into SPARQL/SQL queries and AIR policies

$$\frac{N}{N_{correct} + 1.5 * N_{fp} + 2 * N_{fn}}$$

where,
N is total number of queries
$N_{correct}$ is the number of queries correctly classified
N_{fp} is the number of queries incorrectly classified as violating policy
N_{fn} is the number of queries incorrectly classified as conforming to policy

Policy Assurance metric

Next steps

- Extend Justification UI to provide more relevant explanations
- Develop methodology to generate more specific AIR rules from abstract policies and often ambiguous policies
- Policy development toolkit

```
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix math: <http://www.w3.org/2000/10/swap/math#>.
@prefix s: <http://dig.csail.mit.edu/2009/IARPA-PIR/sparql#> .
@prefix : <http://dig.csail.mit.edu/2009/IARPA-PIR/query1#> .

:Query-5 a s:Select;
   s:cardinality :ALL;
   s:POSList [
      s:variable :S;
      s:variable :N;
      s:variable :ID;
    ];
   s:WhereClause  :WHERE.

   :WHERE a s:DefaultGraphPattern;
      s:TriplePattern { :S <http://xmlns.com/foaf/0.1/age> :A };
      s:TriplePattern  { :S <http://xmlns.com/foaf/0.1/openid> :ID };
      s:Filter [
         a s:ComparatorExpression;
         s:TriplePattern { :A s:BooleanGT "18"^^xsd:integer }
       ];
      s:OptionalGraphPattern [
            s:TriplePattern  { :S <http://xmlns.com/foaf/0.1/ssn> :N };
       ].

#ends
```

SPARQL query in N3

```
:SSN_RULE1 a air:BeliefRule;
      air:label "SSN policy rule1";
      air:pattern {
       :Q a s:Select;
         s:POSList :P;
          s:WhereClause :W.
      };
       air:description (:Q " is a SPARQL query");
       air:rule :SSN_RULE2, :SSN_RULE3, :SSN_RULE4, :SSN_RULE5.

  :SSN_RULE3 a air:BeliefRule;
        air:label "SSN policy rule3";
        air:pattern {
              :P s:variable :V
              :W s:TriplePattern :T.
              :T log:includes { :X <http://xmlns.com/foaf/0.1/ssn> :V }
         };
         air:description ("The query, " :Q  ", uses SSN values in the
  where clause and retrieves SSN values");
         air:assert { :Q air:non-compliant-with :SSNPolicy }.
```

Part of an AIR policy

Clarifications

- Policy compliance separate from PIR protocol (yes)
  - Client -> Query - > Server -> Result -> Client
  - Client -> Query(ies) -> Compliance Checker -> Result -> Client
- Compliance checking (logging)
  - A query at a time, checker responsible for logging queries
  - Client/trusted third party logs queries and sends them to checker
- Compliance checker has access to database/query results (??)
  - E.g. policy - Access to data of minors is prohibited
- Compliance checker has access to client credentials (??)
  - different classes of clients have different policies associated with them

References

- Policy Assurance for PIR Queries, http://dig.csail.mit.edu/2009/IARPA-PIR/
- TAMI project, http://dig.csail.mit.edu/TAMI
- Tabulator extension, http://dig.csail.mit.edu/2007/tab/
- AIR specifications, http://dig.csail.mit.edu/TAMI/2008/12/AIR
- Paper on AIR, http://dig.csail.mit.edu/2008/Papers/IEEEPolicy
- SPASQL, http://www.w3.org/2006/Talks/0518-SPASQL/