Chapter 10

Explaining Assessment Results

The accuracy of information quality assessment results is often uncertain due to the limited availability of quality indicators and due to the uncertain quality of the quality indicators themselves. Therefore, the user's final decision whether to trust or distrust assessment results depends on her understanding of the assessment metrics and quality indicators that were used in the assessment process. Information systems can support users in this trust decision by providing explanations *why* information satisfies a given filtering policy.

Making information filtering decisions comprehensible and traceable requires diverse forms of explanations. The content of suitable explanations depends on the assessment metrics that are used within a policy and on the current task of the user. For less important tasks, the user will be contented with short, simple to comprehend explanations. For others, more important tasks the user will require explanations to contain detailed information about the assessment process and the quality indicators that were used in the process.

Explanations for Rating-Based Metrics. The accuracy of rating-based assessment metrics depends on the the quality of the ratings that are used in the calculation as well as on the scoring algorithm. Ratings might be subjective and raters may try to influence rating systems by providing unfair ratings. Therefore, explanations for rating-based assessment metrics should contain the ratings that were used in the evaluation and explain the calculation steps of the scoring algorithm. More detailed explanations might provide provenance information about ratings and background information about raters.

Explanations for Context-Based Metrics. Context-based assessment metrics rely on meta-information about the circumstances in which information has been claimed. An explanation for a policy that relies on provenance information should list the information providers. The explanation might also contain additional background information about information providers in order to support information consumers in judging their trustworthiness.

Explanations for Content-Based Metrics detail why information content itself satisfies the requirements of an assessment metric. For instance, an explanation for a statistical metric should contain the data that was used in the calculation and describe the calculation process. An explanation for a text analysis method might list relevant keywords and explain how the overall score for a text was calculated.

The WIQA framework can generate explanations why an accepted triple satisfies a WIQA-PL policy. The WIQA framework can produce two types of explanations: Textual and RDF explanations. RDF explanations consist of an RDF graph and may be used by applications for further processing. Textual explanations are aimed at direct human consumption. They consist of natural language text fragments.

In order to provide a high degree of flexibility, the WIQA framework combines two explanation generation mechanisms: First, a template mechanism is used to generate the parts of an explanation which explain why constraints that are expressed as graph patterns are satisfied. Afterwards, the template generated explanation parts are supplemented with custom explanation parts that explain why constraints that are expressed using WIQA extension functions are satisfied.

Figure 10.1 shows a visualization of a textual explanation. The explanation details why a triple fulfills the policy "Accept only information that has been asserted by analysts who have received at least 3 positive ratings". Lines 11-15 explain why Peter Smith is considered to be an analyst. Lines 16-19 explain why Peter Smith satisfies the second part of the policy by listing all raters who have rated him positive.

The content of the "because"-part of the explanation is defined using *explanation templates*. When a user requests an explanation why an accepted triple fulfills the policy, these templates are instantiated with variable bindings from the matching solutions that led to the acceptance of the triple. Chapter 10.1 will explain the template mechanism in detail.

Extension functions may conduct complex calculations and may retrieve additional information from the graph set. In order to make their calculations comprehensible, extension functions can generate custom, function-specific explanations. Chapter 10.2 describes how extension function generated and template generated explanations are combined and discusses the

```
The triple:

  Siemens AG has positive analyst report: "As Siemens agrees
  partnership with Novell unit SUSE ..."

fulfills the policy:

  Accept only information that has been asserted by
  analysts who have received at least 3 positive ratings.

because:

  it was asserted by Peter Smith and
     - Deutsche Bank claims that Peter Smith is an analyst.
     - Financial Times claims that Peter Smith is an analyst.
  Peter Smith has received positive ratings from
     - Mark Scott who works for Siemens.
     - David Brown who works for Intel.
     - John Maynard who works for Financial Times.

```

Figure 10.1: Example explanation.

custom explanations that are generated by the `wiqa:MorePositiveRatings` and the `wiqa:TidalTrust` extension functions.

10.1 Explaining Pattern Matches

WIQA-PL uses a template mechanism to define the content and the structure of explanations. When a user requests an explanation why an accepted triple fulfills the policy, the explanation templates are instantiated with variable bindings from the matching solutions that led to the acceptance of the triple. This section describes the WIQA explanation template mechanism. As a running example, it is explained how the example explanation shown in Figure 10.1 is generated.

Technically, WIQA textual explanations consist of a set of explanation parts. Each explanation part is a tuple $(fragments, children, details)$, where $fragments$ is an ordered list of RDF nodes (usually literals). $fragments$ is created by instantiating an explanation template. *children* is a set of explanation parts which are displayed as children of the current explanation part. *details* may contain an explanation part which contains additional details about the content of $fragments$. As the *details* part may have child parts of its own, this mechanism allows explanations to be divided into different levels of abstraction.

The content of *fragments* is specified by an *explanation template*. Explanation templates are defined within the pattern clause of a WIQA-PL policy. Figure 10.2 shows the WIQA-PL grammar for defining explanation templates. An explanation template consists of an ordered list of literals, variables, and `ExtensionFunctionURI`s. A template is instantiated by replacing the variables within the template with their values from the set of matching solutions that led to the acceptance of the triple.

The structure of an explanation is determined by the position of the explanation templates within the pattern clause of a WIQA-PL policy.

```
1. ExplanationClause   ::= 'EXPL' ExplanationTemplate '.'
2. ExplanationTemplate ::= ( Literal | Variable | ExtensionFunctionURI )+
```

Figure 10.2: EBNF grammar of the WIQA-PL explanation clause.

Figure 10.3 shows the WIQA-PL policy “Accept only information that has been asserted by analysts who have received at least 3 positive ratings”. The pattern clause contains the four explanation templates (lines 9, 17-18, 23, 27) which were used to generate the example explanation shown in Figure 10.1. A WIQA policy suite containing explanation templates for all example policies from chapter 9 is available on the WIQA website[1].

A WIQA explanation is generated in a two step process: First, graph patterns and explanation templates are arranged into a graph pattern tree. Afterwards, the matching solutions that led to the acceptance of the triple are matched against the explanation templates. This two-step process is neccessary to arrange the table-like solution set into a tree-like explanation. Both steps are described below.

10.1.1 Building the Graph Pattern Tree

Graph patterns share variables and contain variables that refer to the `GRAPH ?GRAPH {?SUBJ ?PRED ?OBJ}` root pattern which is added in the matching process to the set of graph patterns given by the policy (see Section 9.3). For example, the graph pattern in lines 6-9 of Figure 10.3 shares the variable `?GRAPH` with the root pattern and the variable `?authority` with the pattern in lines 11-12 and the pattern in lines 20-23.

The structure of the graph pattern tree is determined by the relations of graph patterns to each other through shared variables. Starting from the

[1] http://www.wiwiss.fu-berlin.de/suhl/bizer/wiqa/financialscenario/FinancialScenarioPolicies.wiqa (retrieved 09/25/2006)

```
NAME "Asserted by analysts with at least 3 positive ratings."
DESCRIPTION "Accept only information that has been asserted by
         analysts who have received at least 3 positive ratings."
PATTERNS {

  GRAPH fd:GraphFromAggregator
        { ?GRAPH swp:assertedBy ?warrant .
          ?warrant swp:authority ?authority .
          EXPL "it was asserted by " ?authority " and " . }

  GRAPH ?graph2
        { ?authority rdf:type fin:Analyst . }

  GRAPH fd:GraphFromAggregator
        { ?graph2 swp:assertedBy ?warrant2 .
          ?warrant2 swp:authority ?authority2 .
          EXPL ?authority2 " claims that " ?authority
               " is an analyst." . }

  GRAPH ANY
        { ?rater fin:positiveRating ?authority .
          FILTER (wiqa:count(?rater) > 2) .
          EXPL ?authority "has received positive ratings from" . }

  GRAPH fd:BackgroundInformation
        { ?rater fin:affiliation ?company .
          EXPL ?rater "who works for" ?company . }
  }
```

Figure 10.3: WIQA-PL policy including explanation templates.

root pattern, graph patterns are arranged into a graph pattern tree by the following rule: A graph pattern B becomes a child of another graph pattern A if both patterns share at least one variable, but not if a variable is already shared by pattern A and its parent or between two ancestors of pattern A. The second part of the rule assures that multiple graph patterns that share the same variable with a single parent pattern only become children of this pattern and do not appear a second time in the tree as children of each other.

Applying this rule to the graph pattern set given by our example policy results in the graph pattern tree shown in Figure 10.4. The root pattern shares the variable `?GRAPH` with `Graph Pattern 1`. `Graph Pattern 1` shares the variable `?authority` with the `Graph Pattern 2` and `Graph Pattern 4`. `Graph Pattern 2` shares the variable `?graph2` with `Graph Pattern 3`, and `Graph Pattern 4` the variable `?rater` with `Graph Pattern 5`. `Graph Pattern 4` is not a child pattern of `Graph Pattern 2` as the variable `?authority` is already shared between

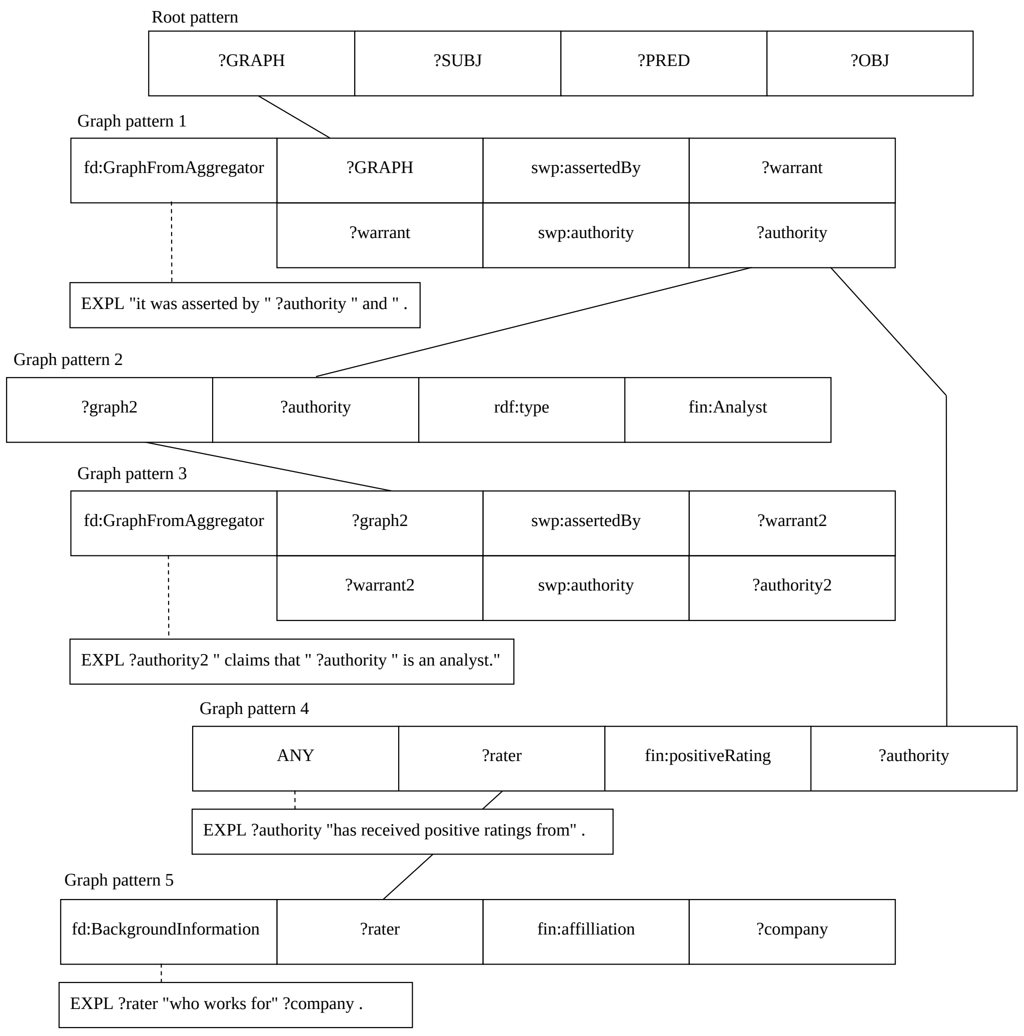

Figure 10.4: Graph pattern tree with attached explanation templates.

pattern one and two.

10.1.2 Instantiating the Graph Pattern Tree

When a user requests an explanation why a triple satisfies a given policy, then the explanation templates in the graph pattern tree are instantiated with variable bindings from the set of matching solutions that led to the acceptance of the triple. This section describes the algorithm for creating an explanation for a given triple from a graph pattern tree and the solution set that led to the acceptance of the triple. Section 9.3 explains how the solution set that leads to the acceptance of a triple is determined.

Let p be the policy that was applied to filter the graph set GS. Let *tree* be the graph pattern tree for policy p, and let *root* be the root pattern of *tree*. Let t be an accepted triple from GS. Let S be the set of matching solutions which led to the acceptance of the triple t.

Figure 10.5 shows the algorithm that is used by the WIQA engine for creating explanations. `explain(S, root, tree)` creates a set of explanation parts why t matches p. The condition in line 6 of the algorithm checks whether *gp* has an attached explanation template *tpl*. If *gp* has an explanation template, then the set V of all variables that are contained in *tpl* is determined. Lines 8-11 determine all distinct sets of bindings of these variables in the solution set S. A binding is a tuple $(variablename, variablevalue)$. The function `projection(s, V)` determines the set of bindings of all variables in V from solution s. The function `instantiate(tpl, bindingset)` in line 13 instantiates *tpl* with a *bindingset* from *bindingsets* by replacing all variables in *tpl* with their values from *bindingset*.

Lines 14-18 create the child parts of the current explanation part. Line 14 determines all solutions in the solution set S that assign the same values to the variables in V as the current *bindingset*. The function `children(gp, tree)` determines the set of child graph patterns of *gp* from the graph pattern tree *tree*. Lines 16-18 recursively call `explain()` for each child pattern *c*. The resulting explanation parts are added to the set *childparts*. Finally, in line 19, a new explanation part, consisting of the template instance $fragments$ and the set *childparts*, is created and added to the set *parts*.

Lines 22-24 are executed if *gp* does not have an attached explanation template. For each child graph pattern of *gp*, the function `explain()` is recursively called and the resulting set of explanation parts is added to to the set *parts*.

Note that the function `instantiate(tpl, bindingset)` replaces the variables in *tpl* with their values from *bindingset* but does not replace variable values that are URIs with the name or label of the resource that is identified by the URI. URIs are not replaced with labels in order to enable applications to display custom labels for resources and to provide functionality for retrieving background information about resources.

10.2 Explaining Extension Function Results

The extension function mechanism introduced in Chapter 9.6 enables the WIQA framework to be extended with domain-specific assessment metrics. Making the results of extension functions comprehensible often requires ex-

```
explain(S, gp, tree)
  input: S, a solution set
  input: gp, a graph pattern
  input: tree, a graph pattern tree
  parts = {}
  if gp has an explanation template tpl
    V = set of all variables in tpl
    bindingsets = {}
    for each s in S
      bindingsets = bindingsets U { projection(s, V) }
    end for
    for each bindingset in bindingsets
      fragments = instantiate(tpl, bindingset)
      solutiongroup = { s in S where bindingset is subset of s }
      childparts = {}
      for each c in children(gp, tree)
        childparts = childparts U explain(solutiongroup, c, tree)
      end for
      parts = parts U { (fragments, childparts) }
    end for
  else
    for each c in children(gp, tree)
      parts = parts U explain(S, c, tree)
    end for
  end if
  return parts
```

Figure 10.5: Algorithm for generating an explanation from a graph pattern tree and the solution set that led to the acceptance of a triple.

tensive, function-specific explanations. For instance, an explanation for a rating-based extension function should contain a description of the scoring algorithm and should list all ratings that were used in the calculation.

The template mechanism described in the last section is suitable for explaining graph pattern matches, but is often too limited for explaining extension function results. Therefore, WIQA-PL allows template generated explanations to be supplemented with custom, function-specific explanation parts. Function-specific explanations are generated directly by the extension function plug-ins. Each plug-in that wants to provide function-specific explanations has to implement the `returnExplanation()` method (see Section 11.2.1). The returned function-specific explanation has to consist of a tree of explanation parts.

This chapter describes how function-specific explanation parts are combined with template generated parts. Afterwards, the custom explanations that are generated by the `wiqa:MorePositiveRatings` and `wiqa:TidalTrust` extension functions are discussed.

Extension function generated explanation trees are included as branches into the template generated explanation tree. The position, where function generated explanation trees are included, is specified by URI references to extension functions in the explanation template. Figure 10.6 shows the WIQA-PL policy "Only accept information from information providers who have received more positive than negative ratings". The policy uses the `wiqa:MorePositiveRatings` extension function (line 10). Line 5-6 contain an explanation template. The template contains the reference `wiqa:MorePositiveRatings`. Each time the template is instantiated, the `returnExplanation()` method of the `wiqa:MorePositiveRatings` extension function plug-in is called with the values of the current matching solution. The resulting explanation tree is added to the explanation part that is generated by the template.

```
1. NAME "More positive ratings"
2. DESCRIPTION "Only accept information from information providers who
3.              have received more positive than negative ratings."
4. PATTERN
5.      {  EXPL "The information was asserted by "
6.               ?authority " and " wiqa:morePositiveRatings .
7.         GRAPH fd:GraphFromAggregator
8.               { ?GRAPH swp:assertedBy ?warrant .
9.                 ?warrant swp:authority ?authority .
10.                FILTER wiqa:morePositiveRatings(?authority) . }
11.     }
12.
```

Figure 10.6: WIQA-PL policy: Only accept information from information providers who have received more positive than negative ratings.

10.2.1 More Positive Ratings Function

The `wiqa:MorePositiveRatings` extension function, introduced in Chapter 9.6.1, implements a simple rating-based scoring algorithm. The function returns true if the graph set contains more positive than negative ratings for a specific resource, and returns false otherwise.

In order to understand the results of the `wiqa:MorePositiveRatings` function and to be able to assess its accuracy, a user requires to know the sums of positive and negative ratings and the origin of the ratings. Therefore, the `wiqa:MorePositiveRatings` function generates function-specific explanations that consist of three parts: The first part contains the sums of ratings. The second and third part list the information providers that rated the resource positive or negative.

Figure 10.7 shows a visualization of an explanation why a triple satisfies the "More positive ratings" policy shown in Figure 10.6. Line 13 of the explanation is generated by the explanation template from the policy. Line 14-25 are generated by the extension function plug-in. The explanation uses the *details* mechanism to divide the explanation into two different levels of detail: Line 15-16 sum up the positive and negative ratings. Detail 1 (line 18-21) lists all positive ratings. Detail 2 (line 23-25) lists all negative ratings.

An application which displays explanations to the user might only show the main part of an explanation by default, and display the details only if they are explicitly requested by the user.

```
1.  The triple:
2.
3.   Siemens AG has positive analyst report: "As Siemens agrees
4.   partnership with Novell unit SUSE ..."
5.
6.  fulfills the policy:
7.
8.   Only accept information from information providers who
9.   have received more positive than negative ratings.
10.
11. because:
12.
13.  The information was asserted by Peter Smith and
14.  Peter Smith received the following numbers of ratings:
15.        - 3 positive ratings (see detail 1)
16.        - 2 negative ratings (see detail 2)
17.
18.  Detail 1: Peter Smith received positive ratings from:
19.        - John Reynolds
20.        - Mary O'Conner
21.        - Elisa Armstoen
22.
23.  Detail 2: Peter Smith received negative ratings from:
24.        - Dave Berser
25.        - Colin Marwick
26.
```

Figure 10.7: Explanation why a triple matches the policy: Only accept information from information providers who have received more positive than negative ratings.

10.2.2 Tidal Trust Function

The `wiqa:TidalTrust` extension function, introduced in Chapter 9.6.2, implements a rating-based scoring algorithm that takes only ratings from information providers into account who are on the information consumer's web-of-trust [Jen05]. The algorithm operates on a network of ratings in which each node has rated several other nodes. The score for a resource is calculated in a three step process: First, the algorithm determines all minimum length paths in the network between the information consumer (source node) and the resource (sink node). Then, the threshold *max* is set to the maximum strength of these paths. Afterwards, each node on the paths calculates its rating for the sink. A rating is calculated by taking the weighted average of all ratings for the sink from the successors of a node, that are rated above the threshold *max* by the node. Each rating is weighted with the rating of the node for its successor (see Formula 9.15 in Section 9.6.2).

A custom explanation for the `wiqa:TidalTrust` extension function therefore has to explain which ratings were taken into account and describe how the rating for the sink was calculated from these ratings.

The `wiqa:TidalTrust` extension function generates explanations consisting of a summary and three blocks of details. Figure 10.8 show an example explanation for the policy "Only accept information from information providers with a Tidal Trust rating above 5". Lines 13-15 sum up the calculation result. Detail number 1 (lines 19-24) gives an overview of the calculation process. Detail number 2 (lines 26-31) lists all minimum length path from the source to the sink. Detail number 3 (lines 33-46) explains how each score is calculated.

10.3 RDF Explanations

Beside of displaying explanations to the end-user, applications might require to process explanations in other application-specific ways. For instance, applications could use explanations as input for reasoning processes, perform different actions depending on the content of an explanation, or attach explanations as evidence to information that is exchanged with other applications [MdS03].

In order to support these use cases, the WIQA framework can generate pure RDF explanations. An RDF explanation consist of an RDF graph. The content of the graph is specified using a *construct template*. Figure 10.9 shows the grammar for defining construct templates within WIQA-PL policies. A construct template is introduced by the keywords CONSTRUCT

EXPLANATION and consists of a set of *construct triple patterns*.

An RDF explanation is generated by taking each matching solution in the solution set that led to the acceptance of the triple, substituting each variable in the construct template with its value from the matching solution and combining the resulting triples into a single RDF graph.

Figure 10.10 shows a WIQA-PL policy containing a construct template in lines 11-14. The construct template consists of two construct triple patterns. The template generates an RDF explanation containing the author and the publication date of each graph in the graph set to be filtered that is published after January 1st, 2006. Figure 10.11 shows an example RDF explanation which was generated with the construct template shown in Figure 10.10.

```
The Triple:

  Intel Share has discussion forum posting: As we have already seen
  in the past, investing into this company is no good idea.

fulfills the policy:

  Only accept information from information providers with a
  Tidal Trust rating above 5.

because:

   It was asserted by Mark Scott. The WIQA extension function
   Tidal Trust inferred a rating of 6.7 from Chris Bizer for Mark
   Scott (see detail 1).

Details:

Detail 1: The inferred rating arises from the following calculation:
  - The shortest path between Chris Bizer and Mark Scott has length 3.
    There are 4 different paths with that length (see detail 2).
  - The maximum strength of the paths is 6.0. Therefore, ratings
    below 6.0 are ignored.
  - The calculation yielded a result of 6.7 (see detail 3).

Detail 2: Paths from the source to the sink:
  - Chris Bizer -6.0-> Anne Richards -9.0-> John Gevner -9.0->
    Mark Scott (Strength of the path: 6.0)
  - Chris Bizer -9.0-> Siddhartha Kataki -7.0-> Mary Louis -6.0->
    Mark Scott (Strength of the path: 6.0)
  - ...

Detail 3: Chris Bizer -6.7-> Mark Scott was calculated from these
  ratings:
  - Siddhartha Kataki -6.0-> Mark Scott, weighted with Chris Bizer's
    rating of 9.0 for Siddhartha Kataki.
  - Anne Richards -7.8-> Mark Scott, weighted with Chris Bizer's
    rating of 6.0 for Anne Richards.
  - Anne Richards -7.8-> Mark Scott was calculated from these ratings:
    - Mary Louis -6.0-> Mark Scott, weighted with Anne Richards's
      rating of 6.0 for Mary Louis.
    - John Gevner -9.0-> Mark Scott, weighted with Anne Richards's
      rating of 9.0 for John Gevner.
  - ...
  - John Gevner -9.0-> Mark Scott is a direct rating.
  - Mary Louis -6.0-> Mark Scott is a direct rating.
```

Figure 10.8: Explanation why a triple matches the policy: Only accept information from information providers with a Tidal Trust rating above 5 (shortened).

```
RDFExplanationClause ::= 'CONSTRUCT EXPLANATION' ConstructTemplate
ConstructTemplate  ::= '{' ConstructPattern+ '}'
ConstructPattern   ::= URIOrBnodeOrVariableOrReference
                       URIOrBnodeOrVariableOrReference
                       URIOrBnodeOrLiteralOrVariableOrReference '.'
URIOrBnodeOrVariableOrReference
                   ::= URI | Bnode | Variable | Reference
URIOrBnodeOrLiteralOrVariableOrReference
                   ::= URI | Bnode | Variable | Literal | Reference
```

Figure 10.9: EBNF grammar for defining WIQA-PL construct templates.

```
NAME "Information that has been asserted after January 1st, 2006"
DESCRIPTION "Accept only information that has been asserted
             after January 1st, 2006"
PATTERN
      { GRAPH fd:GraphFromAggregator
          { ?GRAPH swp:assertedBy ?warrant .
            ?warrant swp:authority ?authority .
            ?warrant dc:date ?date .
            FILTER (?date > "2006-01-01"^^xsd:date) }
      }
 CONSTRUCT EXPLANATION
      {     ?GRAPH dc:creator ?authority .
            ?GRAPH dc:date ?date .
      }
```

Figure 10.10: WIQA-PL policy containing a construct template.

```
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix fd: <http://www.fu-berlin/suhl/bizer/exampleDataset> .

fd:GraphFromPeterSmith
             dc:creator <mailto:peterSmith@deutsche-bank.de> ;
             dc:date "2005-11-20T12:40:44"^^xsd:dateTime .

fd:GraphFromMarkScott dc:creator <mailto:mark@scott.com> .
             dc:date "2005-11-20T17:22:10"^^xsd:dateTime .
```

Figure 10.11: Example RDF explanation generated with the construct template shown in Figure 10.10.